NAS1-1858

Old Dominion University Research Foundation

DEPARTMENT OF CIVIL ENGINEERING
COLLEGE OF ENGINEERING & TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

PARALLEL-VECTOR COMPUTATION FOR STRUCTURAL ANALYSIS AND NONLINEAR UNCONSTRAINED OPTIMIZATION PROBLEMS

By

Duc T. Nguyen, Principal Investigator

Final Report
For the period ended June 15, 1990

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Master Contract Agreement NAS1-18584
Task Authorization No. 59
Dr. Jaroslaw Sobieski, Technical Monitor
Interdisciplinary Research Office

September 1990

Old Dominion University Research Foundation is a not-for-profit corporation closely affiliated with Old Dominion University and serves as the University's fiscal and administrative agent for sponsored programs.

Any questions or comments concerning the material contained in this report should be addressed to:

Executive Director
Old Dominion University Research Foundation
P. O. Box 6369
Norfolk, Virginia 23508-0369

Telephone: (804) 683-4293
Fax Number: (804) 683-5290

DEPARTMENT OF CIVIL ENGINEERING
COLLEGE OF ENGINEERING & TECHNOLOGY
OLD DOMINION UNIVERSITY
NORFOLK, VIRGINIA 23529

PARALLEL-VECTOR COMPUTATION FOR STRUCTURAL ANALYSIS AND NONLINEAR UNCONSTRAINED OPTIMIZATION PROBLEMS

By

Duc T. Nguyen, Principal Investigator

Final Report
For the period ended June 15, 1990

Prepared for
National Aeronautics and Space Administration
Langley Research Center
Hampton, Virginia 23665

Under
Master Contract Agreement NAS1-18584
Task Authorization No. 59
Dr. Jaroslaw Sobieski, Technical Monitor
Interdisciplinary Research Office

Submitted by the
Old Dominion University Research Foundation
P.O. Box 6369
Norfolk, Virginia 23508-0369

September 1990

TABLE OF CONTENTS

I. OBJECTIVE AND MOTIVATION

Practical engineering application can often be formulated in the form of a constrained optimization problem. There are several solution algorithms for solving a constrained optimization problem. One approach is to convert a constrained problem into a series of unconstrained problems. Furthermore, unconstrained solution algorithms can be used as part of the constrained solution algorithms. Structural optimization is an iterative process where one starts with an initial design, a finite element structure analysis is then performed to calculate the response of the system (such as displacements, stresses, eigenvalues, etc.). Based upon the sensitivity information on the objective and constraint functions, an optimizer such as ADS (Ref. 1) or IDESIGN (Ref. 2), can be used to find the new, improved design. The entire process can be summarized in Figure 1.

From Figure 1, it can be identified that a major computational effort occurs in the structural analysis phase to find the static solution, the eigenvalue solution, and/or the dynamic solution of the governing equations of motion.

For the structural analysis phase, the equation solver for the system of simultaneous, linear equations plays a key role since it is needed for either static, or eigenvalue, or dynamic analysis. The equation solver is also needed for the sensitivity analysis and optimization phase.

For practical, large-scale structural analysis-synthesis applications, computational time can be excessively large. Thus, it is necessary to have a new structural analysis-synthesis code which employs new solution algorithms to exploit both parallel and vector capabilities offered by modern, high performance computers (available at NASA Langley Research Center) such as the Convex, Cray-2 and Cray-YMP computers.

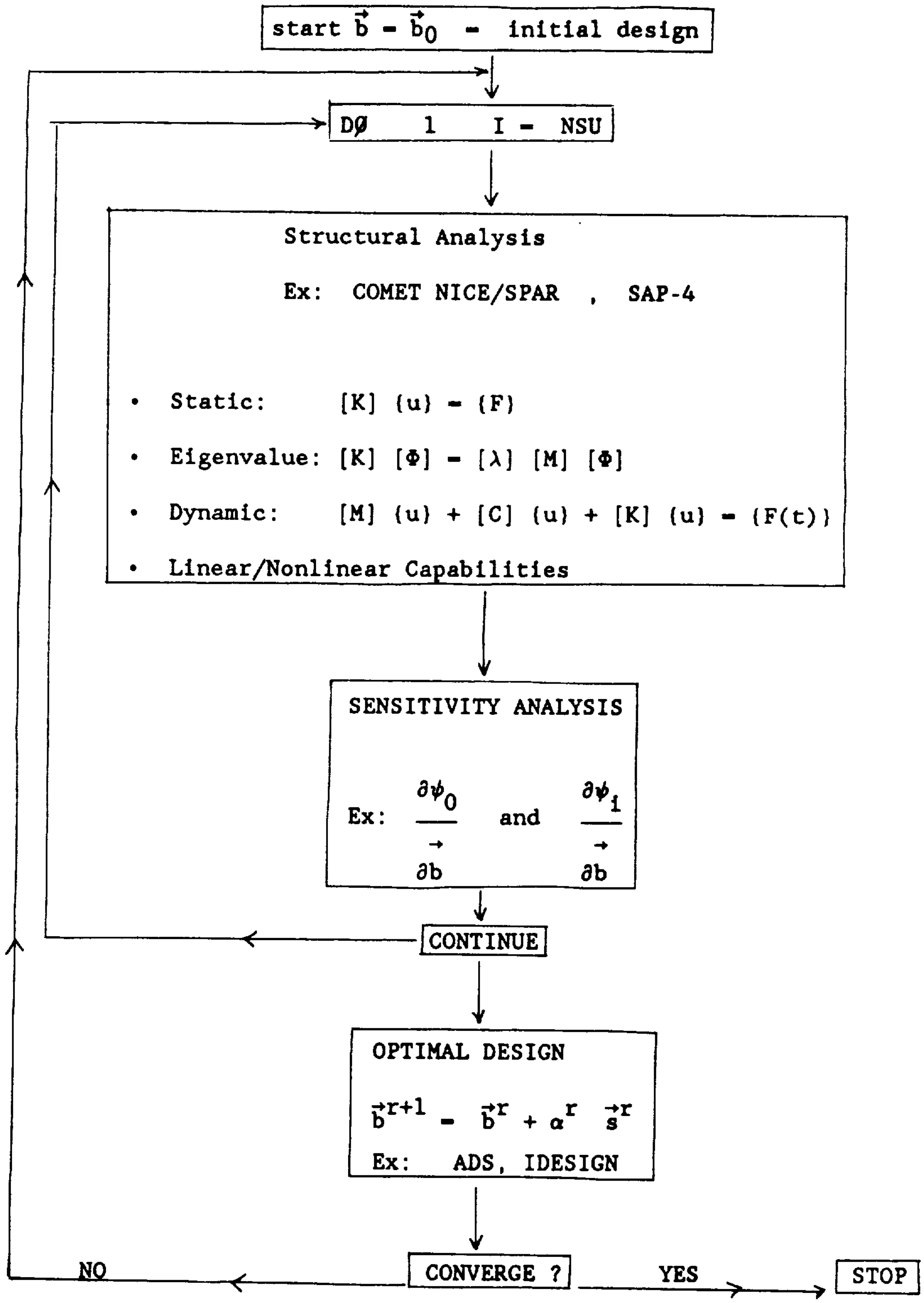


Figure 1. A General Flow Chart for Structural Optimization.

The objectives of this research project are, therefore, to incorporate the latest development in the parallel-vector equation solver, PVSOLVE (See Appendix A) into the widely popular finite-element production code, such as the SAP-4 (See Appendix D). Furthermore, several nonlinear unconstrained optimization subroutines have also been developed and tested under a parallel computer environment. These unconstrained optimization subroutines are not only useful in their own right, but they can also be incorporated into a more popular constrained optimization code, such as ADS (Ref. 1).

II. STRUCTURAL ANALYSIS

There are many finite-element based structural analysis codes available in the literature. The SAP-4 code (See Appendix D) has been selected in this research project due to the following four main reasons.

1. SAP-4 code is in the public domain. The FORTRAN source code is available to all users and the code can be modified to incorporate new numerical algorithms.
2. SAP-4 code has a good number of finite element libraries and has options for static, eigenvalue, and dynamic analysis.
3. Both the in-core, and out-of-core solution options are available in SAP-4. Thus, large scale finite-element models can be handled by the code.
4. SAP-4 code has been written in a modular fashion, thus new capabilities can be added to the code without too much effort.

2.1 General Description of SAP-4 Code

SAP-4 is a general purpose, finite-element code which has been developed and widely used in the industries, government laboratories, and academia in the 1970's. SAP-4 finite element library includes:

- Three-dimensional truss element
- Three-dimensional beam element
- Plane stress, plane strain and axisymmetric elements
- Three-dimensional solid element
- Thick shell element
- Thin plate and shell element
- Boundary element
- Pipe element

The following linear finite element analysis capabilities of SAP-4 are available

- Static analysis
- Calculation of frequencies and mode shapes
- Dynamic analysis

For a more detailed description of SAP-4 code, a complete SAP-4 manual is given in Appendix D.

2.2 Modification of SAP-4 to PV-SAP (Parallel-Vector Structural Analysis Program)

In order to incorporate the newly developed Parallel-Vector equation SOLVEr, PVSOLVE (See Appendix A) into the SAP-4 code, the following modifications have been made in the SAP-4 code:

- Calculating the address of the diagonal terms of the (one-dimensional) coefficient stiffness matrix.

- Assembling the global coefficient stiffness matrix in a row-oriented, variable band fashion.
- Solving the system of simultaneous linear equations by PVSOLVE.

The complete listing of the new code, PV-SAP, is given in APPENDIX E.

2.3 Static Application of PV-SAP Code

In order to evaluate the performance of the new PV-SAP code as compared to the original SAP-4 code, the following examples have been considered.

Example 1: Two-Hundred Bay, Ten Story (2D) Truss Structure

The geometrical pattern as well as the load of this structure is shown in Figure 2. Computational time (using subroutine timef) for the new PV-SAP code, and the original SAP-4 code (using the Cray-2 super computer at NASA Langley Research Center) is shown in Table 1. A parallel speed-up factor of 3.59 (which corresponds to a total equation solver time of 1.05 seconds) was achieved in this example when 4 Cray-2 processors were used. Furthermore, when one processor was used, the new code PV-SAP used only 3.76 seconds as compared to 15.47 seconds from the original SAP-4 code. This significant reduction in time (even for one processor) is due to the fact that the new equation solver (See Appendix A) in PV-SAP has utilized the loop-unrolling technique for better vector speed. In this example, PV-SAP code is 14.75 times faster than the original SAP-4 code.

Example 2: One Hundred Fifty Bay, Ten Story (2D) Frame Structure

The geometrical pattern and the load of this structure is shown in Figure 3. Computational time (using subroutine timef) for the new PV-SAP

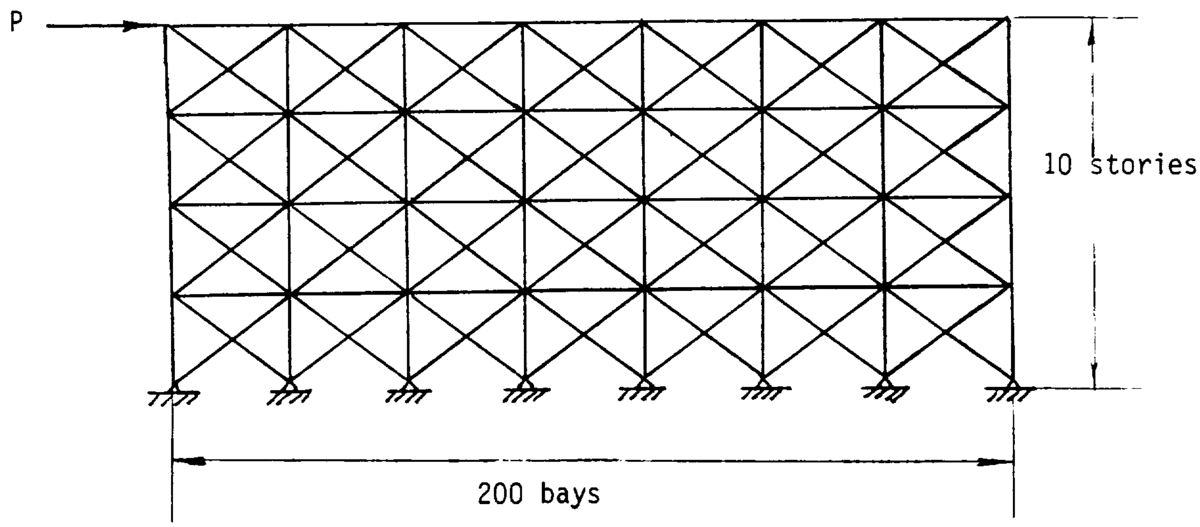


Figure 2: Geometrical Pattern and Loads of Example 1

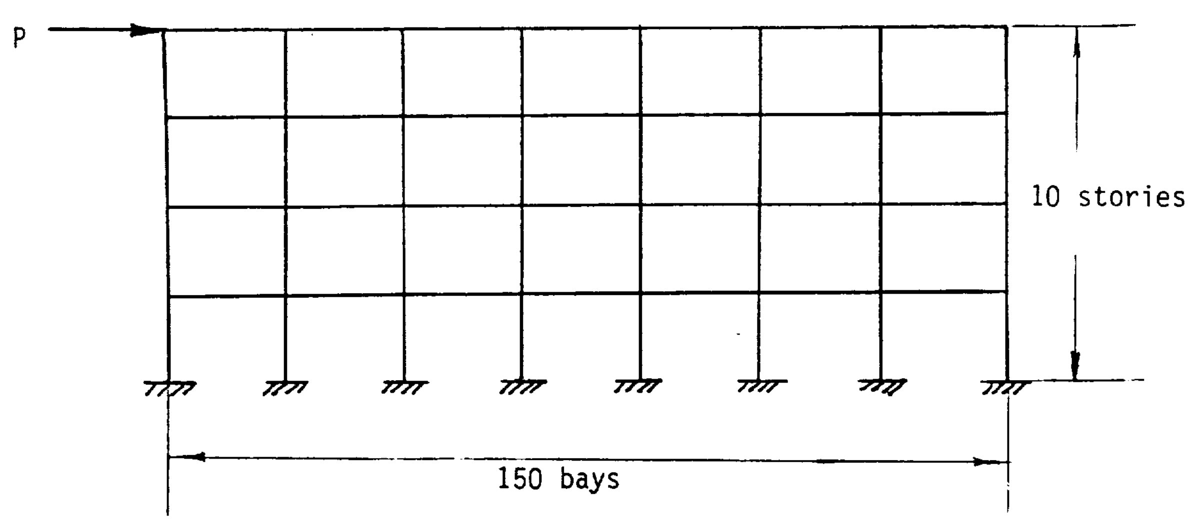


Figure 3: Geometrical Pattern and Loads of Example 2

Table 1. Performance of PV-SAP vs. SAP-4 Code on Example 1.

No. of Processors	Total Equation Solver Time (using seconds)	Speed Up (using seconds)	Total Equation Solver Time (using timef)	Speed Up (using timef)
1	3.82 (SAP-4 = 12.48)	1.00	3.762 (SAP-4=15.469)	1.000
2	2.04	1.87	1.945	1.934
3	1.49	2.56	N/A	N/A
4	1.23	3.11	1.049	3.586

Table 2. Performance of PV-SAP vs. SAP-4 Code on Example 2.

No. of Processors	Total Equation Solver Time (using seconds)	Speed Up (using seconds)	Total Equation Solver Time (using timef)	Speed Up (using timef)
1	5.24 (SAP-4 = 16.47)	1.00	5.123 (SAP-4=15.469)	1.00
2	2.86	1.83	2.657	1.928
3	2.02	2.59	N/A	N/A
4	1.81	2.90	1.414	3.623

code, and the original SAP-4 code (using the Cray-2 supercomputer at NASA Langley Research Center) is shown in Table 2. In this example, PV-SAP code is 10.94 times faster than the original SAP-4 code.

Example 3: Two Hundred Eighty Bay, Five Story (2D) Frame Structure

The geometrical pattern and the load of this structure is the same as shown in Figure 3, except for the number of bays and the number of stories.

Computational time (using subroutine timef) for the new PV-SAP code, and the original SAP-4 code (using the Cray-2 super computer at NASA Langley Research Center) is shown in Table 3. In this example, PV-SAP code is 15.65 times faster than the original SAP-4 code.

III. STRUCTURAL OPTIMIZATION

The purpose of Design Optimization is searching for the best solution with a limited resource. In many engineering applications, design optimization starts with formulating the problem and follows by solving it using a mathematical programming technique. The general formulation of a design optimization problem is given as

$$\min_{b \in R^n} f(b,x) \tag{3.1}$$

subject to

$$g_j(b,x) \leq 0, \; j = 1, \ldots m \tag{3.2}$$

$$h_k(b,x) = 0, \; k = 1, \ldots l \tag{3.3}$$

$$b_{il} \leq b_i \leq b_{iu}, \; i = 1, \ldots n \tag{3.4}$$

Table 3. Performance of PV-SAP vs. SAP-4 Code on Example 3.

No. of Processors	Total Equation Solver Time (using seconds)	Speed Up (using seconds)	Total Equation Solver Time (using timef)	Speed Up (using timef)
1	14.19 (SAP-4 = 56.74)	1.00	13.684 (SAP-4=58.592)	1.000
2	7.24	1.96	6.995	1.956
3	5.31	2.67	N/A	N/A
4	4.62	3.07	3.743	3.660

where b and x are the design and state variables, respectively. Furthermore, the equality constraints $h_k(b,x) = 0$, may include state equations that yield the solution of state variables.

The above design optimization problem is generally nonlinear and it can only be solved numerically. One class of numerical schemes is called the direct search technique, which iteratively looks for a better design in the design space and stops only when certain convergence criteria are satisfied. In other words, in each iteration, the technique finds a better design as

$$x_{new} = x_{old} + \alpha P \tag{3.5}$$

where α is a scalar quantity defined as the step size and P is a vector defining a search direction to improve the solution. Usually, the search direction, P, is the solution of a subproblem which is obtained by linearizing the optimal design problem, Eqs. (3.1)-(3.4). The subproblem can be either unconstrained or constrained.

The software package, Automated Design Synthesis, or ADS (Ref. 1), can be a good candidate for solution of the optimal design problem. ADS is a general-purpose optimization package that offers various algorithms to find the optimal solution. An ADS user can select one of the nine strategy options to formulate a subproblem which subsequently can be solved by one of the five optimizers, depending upon the formulation of the subproblem. Among the five optimizers, Fletcher-Reeves algorithm, Davidon-Fletcher-Powell and Braydon-Fletcher-Goldfarb-Shanno variable metric methods are used for unconstrained subproblems and two versions of feasible direction methods for constrained subproblems.

Once the search direction, P, is found, a proper step size, α, in Eq. (3.5) is computed in order to completely define the new design x_{new}. The best α is determined in such a way that the new design can reduce the objective, as well as correct the constraint violations. Determination of a proper α is usually the most time consuming process in a design optimization algorithm, because it requires many function analyses. To determine α, ADS provides eight different one-dimensional search algorithms, among which five find the minimum of an unconstrained function and three find the minimum of a constrained function.

It should be noted that an ADS user should select a design optimization algorithm which is consistent with the strategy options, the optimizers and the one-dimensional search algorithms. That is, for example, an optimizer for an unconstrained problem should be selected in ADS if an unconstrained subproblem is formulated by the strategy option selected.

The ADS is a collection of subroutines. The ADS can be invoked by calling the subroutine ADS, as follows: Call ADS (INFO, ISTRAT, IOPT, IONED, IPRINT, IGRAD, NDV, NCON, X, VLB, VUB, OBJ, G, IDG, NGT, IC, DF, A, NRA, NCOLA, WK, NRWK, IWK, NRIWK), where the integer parameters, ISTRAT, IOPT, IONED and IPRINT are defined as:

ISTRAT: Optimization strategy to be used.

IOPT: Optimization to be used.

IONED: One-dimensional search algorithm to be used.

IPRINT: A four-digit print control.

An ADS user has the option to either require ADS to calculate function gradients using the finite difference method or to provide function gradients himself. The user should use the arrays DF and A in subroutine ADS to store the gradient information. Furthermore, since the active constraint strategy is employed in ADS, the user should only provide the gradients of constraints that are active. The active constraints can be identified by the array IC. Application examples are given in the ADS manual to demonstrate how to use the ADS software package. Other important aspects such as restarting the code and redefining control parameters in ADS are also detailed in the ADS manual.

3.1 Parallel Golden Block Search Technique

In this research work, a parallel version of the Golden Block Search technique has been developed for determining the step size α in Eq. (3.5). Theoretical development of the Golden Block Search technique [3] is summarized in the following paragraphs:

- The Golden Section method is based on the Fibonacci sequence, which is defined as

$$F_0 = 1 \; ; \quad F_1 = 1 \; ; \qquad \begin{cases} F_n = F_{n-1} + F_{n-2} \\ \text{where } n = 2,3,4 \ldots \end{cases}$$

  with the properties

$$\left.\frac{F_n}{F_{n-1}}\right|_{n \to \infty} = \tau = \frac{1}{2}(1 + \sqrt{5}) \approx 1.618 = \text{golden ratio}$$

- The Fibonacci Sequence is a special case of the Arriel Sequence

$$A_k^0 = 0 \; ; \quad A_k^1 = 1 \; ; \qquad \begin{cases} A_k^n = k\,(A_k^{n-1} + A_k^{n-2}) \\ \text{where } n = 2,3,4 \ldots \end{cases} \tag{3.6}$$

Thus, when $k = 1$, then the Arriel Sequence will become the Fibonacci Sequence

- In order to apply the Arriel Sequence to modify the Golden Search technique, we assume:

$$\frac{A_k^{n+2}}{A_k^{n+1}} = \frac{A_k^{n+1}}{A_k^{n}} = \tau_k \quad \text{as} \qquad n \to \infty \tag{3.7}$$

and try to derive the condition for which τ_k (refer to Eq. 3.7) needs to be satisfied.

- Derivation of a formula for τ_k

Multiplying $\left(\dfrac{A_k^{n+1}}{A_k^{\,n}}\right)$ to both sides of Eq. (3.7)

$$\frac{A_k^{n+2}}{A_k^{\,n+1}} * \frac{A_k^{n+1}}{A_k^{n}} = \left(\frac{A_k^{n+1}}{A_k^{n}}\right)^2 = (\tau_k)^2 \tag{3.8}$$

From Eq. (3.6), one has: $A_k^{n+2} = k\ (A_k^{n+1} + A_k^{n})$ (3.9)

Substituting Eq. (3.9) into (3.8), one obtains:

$$\frac{k\left(A_k^{n+1} + A_k^{n}\right)}{A_k^{n}} = \tau_k^2 = k\left(\frac{A_k^{n+1}}{A_k^{n}} + 1\right)$$

or

$$\tau_k^2 = k\ (\tau_k + 1) \tag{3.10}$$

Solving the quadratic Eq. (3.10) and using only the positive root, one has:

$$\tau_k = \frac{1}{2} (k + \sqrt{k^2+4k}) \tag{3.11}$$

NOTE: If k = 1, then $\tau_k = \frac{1}{2}(1 + \ 5) = 1.618$ = the standard golden section ratio

The above Golden Block Search Algorithm can be conveniently presented in a form of a step-by-step algorithm (also refer to Figure 4).

<u>Step 1</u>: $d_B^0 = b - a$

<u>Step 2</u>: First block search (for i = 1)

- $\alpha_0^1 = a$
- $\alpha_1^1 = a + (\frac{1}{\tau_k})\, d_B^0$ where $\tau_k = \frac{1}{2} (k + \sqrt{k^2+4k})$
- $\alpha_j^1 = \alpha_{j-2}^1 + (\frac{1}{k})\, d_B^0$ where $j = 2, 3, \ldots, 2k$
- parallel computation for $F(\alpha_j^1)$ where $j = 0, 1, 2, 3, \ldots, 2k$

<u>Step 3</u>: Find the value of α_j^1 which gives the minimum value of F, say

$\alpha_j^1 = \alpha_\ell^1$

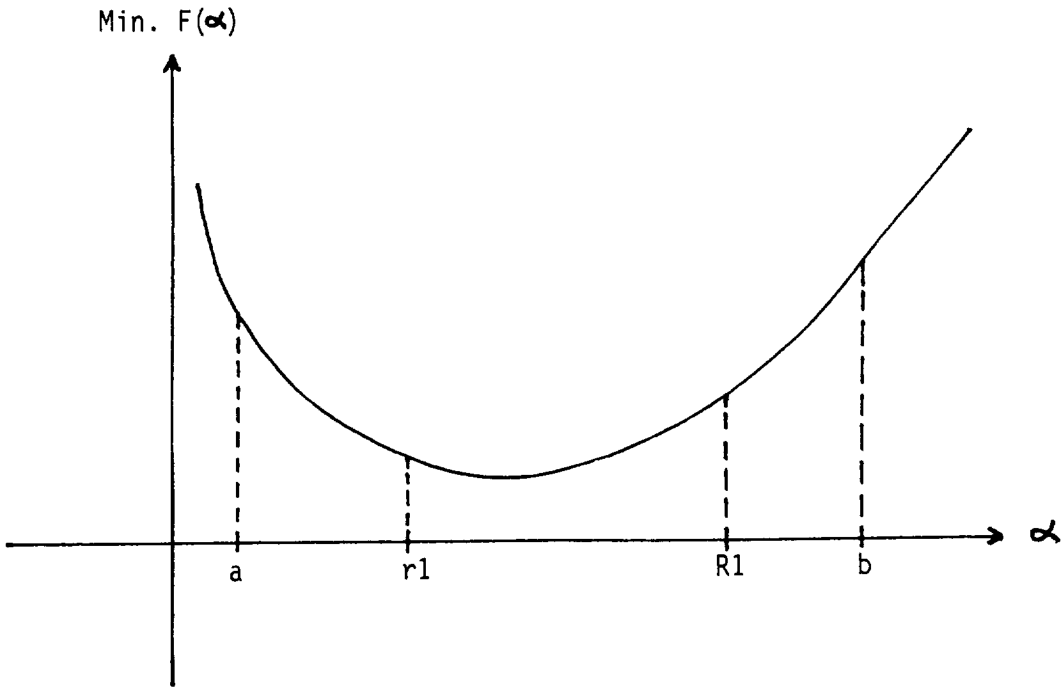


Figure 4: Golden Block Search Algorithm

Step 4: Set $r_1 = \alpha^1_{\ell-1}$ and $R_1 = \alpha^1_{\ell+1}$

Thus $d^1_B = R_1 - r_1 = \dfrac{d^0_B}{k}$

Step 5: Subsequent i^{th} block search (for $i \geq 2$)

$$\alpha^i_0 = r_{i-1}$$

$$\alpha^i_1 = r_{i-1} + \left(\frac{1}{\tau_k}\right)^i \quad d_B{}^0$$

$$\alpha^i_j = \alpha^i_{j-2} + \left(\frac{d^o_B}{k}\right) * (\tau_k)^{1-i} \text{ where } j = 2, 3, \ldots, 2k$$

compute $F(\alpha^i_j)$

Step 6: Return to step 3 if the process does not converge

Based upon the above step-by-step procedure, the parallel golden block search algorithm has been developed, and the complete listing of this subroutine is given in Appendix B.

A Simple Example on Golden Block Search Method

Min $F(x) = 2.0 + e^x - 4x$

use $k = 4$, thus, according to Eq. (3.11), one has

$$\tau_k = \frac{1}{2} \left(4 + \sqrt{16 + 16}\right) = 4.8284271$$

Table 4 indicates that the Golden Block Search method converges in five iterations.

An Example for Parallel Golden Block Search Method

Find t which minimizes the function

$$F(t) = \cos(t) = 1 - \frac{t^2}{2!} + \frac{t^4}{4!} - \frac{t^6}{6!} + \ldots + \frac{t^n}{n!} + \ldots \qquad (3.12)$$

The optimum solution is $t = t^* = \pi$ and $F = F^* = -1.0$.

The following symbols are used in Table 5:

NP - number of processors used

k - the coefficient given in Eq. (3.11)

n - number of terms used in Eq. (3.12)

S - speed up factor

η - efficiency

ε - convergence tolerance

The performance of the Parallel Golden Block Search algorithm is shown in Table 5.

3.2 Parallel-Vector BFGS Method

In these methods, the Hessian rather than its inverse is updated at every iteration. We shall present a method that is most popular and has proved to be most effective in applications. Detailed derivation of the method is given in Gill et al. (also see Reference 2). It is known as the BFGS (Broyden-Fletcher-Goldfarb-Shanno) method described as follows.

Step 1: Estimate an initial design $b^{(0)}$. Choose a symmetric positive definite matrix $H^{(0)}$ as an estimate for the Hessian of the cost function. In

Table 4. Sequential Golden Block Search Example.

Iter. No. F value	j = 0	j = 1	j = 2	j = 3	j = 4	j = 5	j = 6	j = 7	j = 8	j = 9
α =	0.0	0.621320	0.75	1.371320	1.50	2.121320	2.25	2.871320	3.0	
F(α) =	3.0	1.376103	1.11700	0.4553	0.481689	1.856863	2.4877	8.175039	10.0855	
α =	0.75	0.8486	0.9053	1.0340	1.0607	1.1893	1.2160	1.3447	1.3713	1.50
F(α) =	1.1170	0.8930	0.8514	0.6597	0.6456	0.5276	0.5097	0.4582	0.4553	0.481689
α =	1.3447	1.3713	1.3768	1.4035	1.4090	1.4357	1.4412	1.4678	1.4733	1.50
F(α) =	0.4582	0.4553	0.4550	0.4554	0.4559	0.4598	0.4609	0.4685	0.4770	0.4817
α =	1.3713	1.3768	1.3779	1.3835	1.3846	1.3902	1.3913	1.3968	1.3980	1.4035
F(α) =	0.4553	0.4550	0.45496	0.45484	0.45483	0.4585	0.4587	0.45505	0.4551	0.45542
α =	1.3835	1.3846	1.3849	1.3860	1.3863	1.3874	1.3876	1.3888	1.3890	1.3902
F(α) =	0.45484	0.45483	0.45483	0.45482	0.45482	0.45483	0.45483	0.45483	0.45484	0.45485

NOTE: In Reference 2, the author used the standard Golden Section method, and it took 18 iterations to converge to the same tolerance.

Table 5. Parallel Golden Block Search Example.

$n = 600, \qquad \varepsilon = 1.0 \times 10^{-9}$

k = 1

NP	Time (Seconds)	S	η (%)
1	0.3553	1	100.

k = 2

NP	Time (Seconds)	S	η (%)
1	0.3381	1	100.

k = 3

NP	Time (Seconds)	S	η (%)
1	0.3866	1	100.
2	0.2008	1.925	96.25
3	0.13668	2.83	94.30
4	0.12797	3.02	75.60

k = 5

NP	Time (Seconds)	S	η (%)
1	0.48918	1	100
2	0.25147	1.95	97.3
3	0.19397	2.52	84.10
4	0.14565	3.36	84.0

k = 6

NP	Time (Seconds)	S	η (%)
1	0.54002	1.0	100
2	0.27735	1.95	97.4
3	0.18711	2.89	96.2
4	0.14365	3.76	94.0

k = 7

NP	Time (Seconds)	S	η (%)
1	0.57734	1.0	100
2	0.29258	1.973	98.7
3	0.20595	2.8033	93.4
4	0.16478	3.504	87.6

the absence of more information, let $\mathbf{H}^{(0)} = \mathbf{I}$. Choose a convergence parameter ε. Set $k = 0$, and compute the gradient vector as

$c(0) = \nabla f(b^{(0)})$ where f is an objective function.

<u>Step 2</u>: Calculate the norm of the gradient vector as $\|\mathbf{c}^{(k)}\|$. If $\|\mathbf{c}^{(k)}\| < \varepsilon$ then stop the iterative process; otherwise continue.

<u>Step 3</u>: Solve the following linear system of equations to obtain the search direction:

$$\mathbf{H}^{(k)}\mathbf{p}^{(k)} = -\mathbf{c}^{(k)}$$

<u>Step 4</u>: Compute optimum step size $\alpha_k = \alpha$ to minimize $f(\mathbf{b}^{(k)} + \alpha\mathbf{p}^{(k)})$.

<u>Step 5</u>: Update the design as

$$\mathbf{b}^{(k+1)} = \mathbf{b}^{(k)} + \alpha_k\mathbf{p}^{(k)}$$

<u>Step 6</u>: Update the Hessian approximation for the cost function as

$$\mathbf{H}^{(k+1)} = \mathbf{H}^{(k)} + \mathbf{D}^{(k)} + \mathbf{E}^{(k)}$$

where the correction matrices $\mathbf{D}^{(k)}$ and $\mathbf{E}^{(k)}$ are given as

$$D^{(k)} = \frac{y^{(k)}y^{(k)^T}}{(y^{(k)} \cdot s^{(k)})} \; ; \qquad E^{(k)} = \frac{c^{(k)}c^{(k)^T}}{(c^{(k)} \cdot p^{(k)})}$$

with $\quad s^{(k)} = \alpha_k p^{(k)}$ (change in design)

$y(k) = c(k+1) - c(k)$ change in gradient)

$c^{(k+1)} = \nabla f(b^{(k+1)})$

Step 7: Set $k = k + 1$ and go to Step 2.

Notice that the first iteration of the method is the same as that for the steepest descent method.

It can be shown that the BFGS update formula keeps the Hessian approximation positive definite if exact line search is used. This is important to know as the search direction is guaranteed to be that of descent for the cost function only if $H^{(k)}$ is positive definite. In numerical calculation, difficulties can arise because Hessian can become singular or indefinite due to inexact line search and round-off and truncation errors. Therefore, some safeguards against the numerical difficulties must be implemented into computer programs for stable and convergent calculations. Another numerical procedure that is extremely useful is to update decomposed factors (Cholesky factors) of the Hessian rather than the Hessian itself. This way the matrix can be numerically guaranteed to be positive definite.

In this project, parallel-vector implementation of the BFGS method has been achieved by incorporating the mixture of both the direct parallel-vector equation solver (see Appendix A) and the iterative parallel-vector equation solver into Step 3 of the above BFGS process. The complete listing of the parallel BFGS code is given in Appendix C. Table 6 summarizes the performance of the BFGS in a parallel computer environment. In Table 6, systems of 200 and 300 coupled, nonlinear equations have been formulated as the nonlinear, unconstrained optimization problems and were solved by the parallel-vector BFGS method.

Table 6. Performance of the BFGS Method in a Parallel Computer Environment.

Problem Size	Total Timing (Sec.) for BFGS Using Mixed Choleski-Gauss Seidel Method	Number of (Converged) Iterations	BFGS Tolerance	Gauss-Seidel Tolerance	No. of Cray-YMP Processors	Speed Up Factor
200 x 200	42.50	9	0.01	0.00001	1	1.00
200 x 200	26.15	9	0.01	0.00001	2	1.63
200 x 200	14.85	9	0.01	0.00001	4	2.86
300 x 300	102.45	11	0.01	0.00001	1	1.00
300 x 300	58.03	11	0.01	0.00001	2	1.77
300 x 300	31.87	11	0.01	0.00001	4	3.21

IV. CONCLUSIONS AND FUTURE RESEARCH

The fast parallel-vector equation solver (See Appendix A) has been incorporated into a well-known SAP-4 finite element structural analysis code. The new code, PV-SAP, has been tested for static applications. Initial results have indicated that the new code, PV-SAP is 10.94 to 15.65 times faster than the original SAP-4 code when 4 Cray-2 (at NASA Langley Research Center) processors were used.

For the one-dimensional line search problem, the parallel Golden Block Search method has been developed. For a simple tested problem, a speed-up factor of 3.76 was obtained when 4 Cray-2 processors were used.

For the nonlinear unconstrained optimization problem, the parallel-vector version of the BFGS method has been developed. Initial results have indicated that a speed-up factor of 3.21 was obtained when 4 Cray-2 processors were used.

Practical structural optimization problems can usually be formulated in the form of a nonlinear constrained optimization problems. All the results obtained from this research work, however, can be directly used for the next phase of this project. The remaining task which needs to be done is to provide the sensitivity information for PV-SAP since this sensitivity information is needed for many existing optimization packages, such as the ADS in Ref. 1.

ACKNOWLEDGMENT

This research work is supported by the NASA Master Contract NAS1-18584, Task Authorization No. 59. Portions of this work are also supported by the grants from NASA Langley Research Center (NAG-1-858), and the Air Force Office of Scientific Research (F49620-88-C-0053).

REFERENCES

1. Vanderplaats, G.N., Sugimoto, H., and Sprague, C.M. "ADS-1: a new general-purpose optimization program." AIAA Paper No. 83-0831, presented at the AIAA/ASME/ASCE/AHS 24th SDM Conference, Lake Tahoe, California, May 1983.

2. Arora, J.S., Introduction to Optimum Design, McGraw-Hill, Inc., 1989.

3. Fei, J., "Parallel Computing Algorithms," 1985 (in Chinese).

APPENDIX A: NASA Technical Memorandum 102614

NASA Technical Memorandum 102614

A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers

Olaf O. Storaasli, Duc T. Nguyen and Tarun K. Agarwal

NOTICE

FOR EARLY DOMESTIC DISSEMINATION

Because of its significant early commercial potential, this information, which has been developed under a U.S. Government program, is being disseminated within the United States in advance of general publication. This information may be duplicated and used by the recipient with the express limitation that it not be published. Release of this information to other domestic parties by the recipient shall be made subject to these limitations.

Foreign release may be made only with prior NASA approval and appropriate export licenses. This legend shall be marked on any reproduction of this information in whole or in part.

Date for general release: April 30, 1992

April 1990

NASA
National Aeronautics and
Space Administration

Langley Research Center
Hampton, Virginia 23665-5225

A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers

Olaf O. Storaasli
Structural Mechanics Division
NASA Langley Research Center, Hampton, VA 23665-5225

Duc T. Nguyen
Department of Civil Engineering
Old Dominion University, Norfolk, VA 23529-0369

and

Tarun K. Agarwal
Department of Civil Engineering
Old Dominion University, Norfolk, VA 23529-0369

Abstract

A fast, accurate Choleski method for the solution of symmetric systems of linear equations is presented. This direct method is based on a variable-band storage scheme and takes advantage of column heights to reduce the number of operations in the Choleski factorization. The method employs parallel computation in the outermost DO-loop and vector computation via the "loop unrolling" technique in the innermost DO-loop. The method avoids computations with zeros outside the column heights, and as an option, zeros inside the band. The close relationship between Choleski and Gauss elimination methods is examined. The minor changes required to convert the Choleski code to a Gauss code to solve non-positive-definite symmetric systems of equations are identified. The results for two large-scale structural analyses performed on supercomputers, demonstrate the accuracy and speed of the method.

Nomenclature

e_a	error norm for solution residuals
e_s	strain energy error norm
{f}	load vector
hpm	hardware performance monitor (Cray)
i,j,k	DO loop indices
ja	job accounting utility (Cray)
[K]	stiffness matrix
MFLOPS	Million FLoating point OPerations/Second
m_{ij}	multipliers for forward substitution
n	number of equations
NP	number of processors
{R}	error residual for solution: [K] {x} - {f}
RAM	Random Access Memory
SAXPY	Σ ax + y, or scalar * vector + vector
second	CPU time function (Cray)
SRB	space shuttle Solid Rocket Booster
timef	elapsed time function (Cray)
[U]	upper triangular, factored stiffness matrix
u_{ij}	terms of upper-triangular matrix
{x}	static structural displacements

1. Introduction

Since the invention of the first electronic computer by Atanasoff to solve matrix equations of order 29 in 1939[1], researchers in many scientific and engineering disciplines have found their problems invariably reduced to solving systems of simultaneous equations that simulate and predict physical behavior. Currently, the solution of linear systems of equations on advanced parallel-vector computers is a key area of research with applications in many disciplines[2-6]. Structural analysis codes in wide use today were developed on single processor computers and often do not fully exploit the vector or parallel processing capability of modern high-performance computers. To achieve a high level of efficiency on parallel-vector supercomputers, a restructuring of the equation solution procedure and the memory and data management of these structural analysis codes is required. For example, the skyline storage technique used in many sequential structural analysis codes lacks the efficiency of other storage techniques used in the solution of linear systems of equations on vector computers[7-8]. Of equal importance, several parallel equation solvers have been demonstrated to work well for static and dynamic structural analyses, eigenvalue and buckling analyses, sensitivity analysis and structural optimization[9-15]. Since high-performance computers currently have both parallel and vector capability, the algorithms that exploit both will achieve optimal performance for these computers.

Based on favorable experience on sequential computers, a parallel-vector Choleski algorithm using a skyline storage scheme was developed and shown to have excellent parallel performance on a Cray 2 supercomputer as the number of processors increased[16]. However, the skyline scheme was found to prohibit the traditional loop unrolling technique used to optimize vector performance, so a less powerful "vector unrolling" strategy was used.

The present paper describes a new algorithm that overcomes the deficiency of skyline storage by using a variable-band storage scheme. The objective of this paper is to describe this new algorithm for solving matrix equations and to demonstrate its accuracy and speed by solving large-scale structural analysis applications on Cray supercomputers.

Since equation solution algorithms depend on the storage scheme selected, two of the storage schemes used most often are discussed in Section 2 of the present paper. A description of how the basic Choleski method was implemented to achieve both vector and parallel speed is discussed in Section 3. The parallel FORTRAN language, Force[17], used to implement this method, is also discussed in Section 3. The results obtained for two large-scale structural analysis problems to evaluate the performance of the algorithm are discussed in Section 4. The minor changes required to convert this newly-developed code from a Choleski algorithm to a Gauss algorithm for solving non-positive-definite symmetric systems of equations are identified with examples in Appendix A. A description of the input data with a simple example is in Appendix B. A listing of the code and its use, both in a stand-alone mode and in the CSM Testbed[18], is described in Appendix C.

2. Data Storage Schemes

The Choleski method for the solution of simultaneous equations requires the decomposition of the matrix of stiffness coefficients, [K], into an upper-triangular, factored stiffness matrix, [U]. Details of this matrix decomposition are given in Section 3 and Appendix A. Two methods most often used in structural analysis codes to store [U] are the variable-band, and skyline techniques.

For large finite-element applications, the user defines the geometry, finite elements and loads of the finite-element model. The user may use automated algorithms to reorder the resulting stiffness matrix, [K], in the form that is most efficient for the solver. The reverse Cuthill-McKee algorithm[19] reorders the [K] matrix into a near minimum bandwidth, and thus is used for the examples in this paper.

In a row-oriented, variable-bandwidth Choleski approach, the bandwidth of each row of the upper-triangular matrix, [U], is defined as the number of coefficients from a diagonal term to the last non-zero coefficient of the row, excluding the diagonal term. The coefficients of the stiffness matrix for a stiffened panel with a circular cutout (bottom of Fig. 1), are plotted in a variable-band format as shown in Fig. 1.

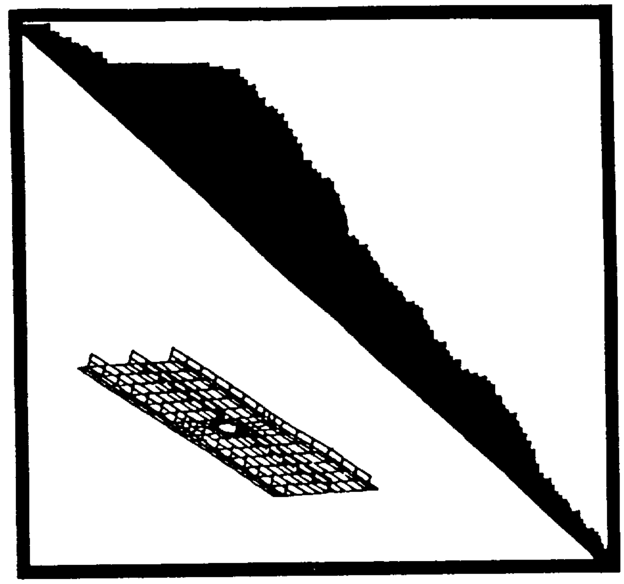


Fig. 1 Variable-band row storage of panel matrix.

The coefficients of the matrix are stored by rows where each row represents a degree of freedom in the finite-element model. The variable-band storage includes all zero coefficients within the so called "profile" which is defined by the ragged right edge of the matrix represented in Fig. 1. Variable-band storage requires less memory than earlier schemes which stored all coefficients within the maximum bandwidth, since earlier schemes stored and operated on many zeros outside the variable-band profile.

The same panel stiffness matrix is stored by columns in the skyline format, like skyscrapers, in Fig. 2 from each diagonal coefficient up to the last nonzero directly above it.

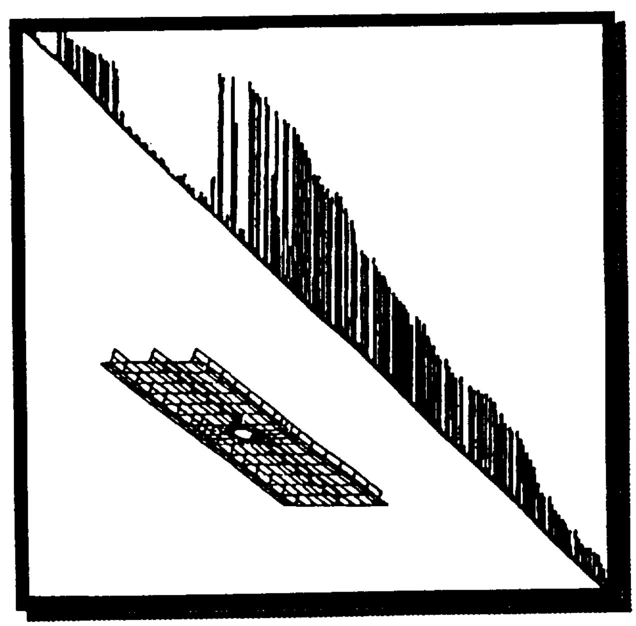


Fig. 2 Skyline column storage of panel matrix.

In this column-oriented storage scheme, the column height is defined as the number of coefficients from a diagonal coefficient to the last nonzero coefficient in the same column, excluding the diagonal coefficient, as shown in Fig. 2. This skyline format requires fewer coefficients to store and operate on during equation solution as indicated by the many zeros (white spaces) in Fig. 2. The panel example is used for illustrative purposes only, as in many applications, the reduction in storage offered by the skyline approach is not so pronounced.

Factorization of a matrix using skyline storage has the advantage that calculations with zeros outside the skyline need not be performed since zeros remain in these locations after factorization. Although the skyline method has the advantage of minimizing the storage and number of operations required on sequential computers, it cannot achieve optimal vector speed on high-performance computers since it cannot use efficient SAXPY operations (i.e., Σ ax + y, or scalar * vector + vector). SAXPY operations achieve optimal performance on vector computers since they continually stream operations to separate add and multiply units which can operate simultaneously.

To compare the storage schemes in detail, the location of the coefficients in the upper half of a 9x9 symmetric stiffness matrix are shown in Fig. 3 as a simple illustrative example.

$$
\begin{array}{cl}
\left[\begin{array}{ccccccccc}
1 & 2 & 0 & 4 & & & & & \\
 & 5 & 6 & 7 & 8 & 9 & & & \\
 & & 10 & 11 & 0 & 0 & & & \\
 & & & 14 & 15 & 16 & 0 & 0 & 19 \\
 & & & & 20 & 21 & 22 & 0 & 24 \\
 & & & & & 25 & 26 & 0 & 28 \\
 & & & & & & 29 & 30 & 31 \\
 & & & & & & & 32 & 33 \\
 & & & & & & & & 34
\end{array}\right]
&
\begin{array}{l}
\\
\leftarrow k=2 \\
\\
\\
\leftarrow k=5 \\
\leftarrow i=6 \\
\\
\\
\\
\end{array}
\end{array}
$$

Fig. 3 Variable-band storage of stiffness matrix.

The non-zero integers in Fig. 3 are the index (location) of each stiffness coefficient stored contiguously in a one-dimensional array. The 34 matrix coefficients are numbered row-wise according to a variable-band storage scheme, where for illustrative purposes, the seven zeros are stored within five of the rows. The skyline storage scheme requires only 29 locations to store the same matrix, since the five zeros in columns 3, 7 and 8 in Fig. 3 fall outside the skyline and need not be stored. The two zeros in row 3 must be stored in both the variable-band and skyline storage schemes since they may become non-zero during factorization. The bandwidth of row 2 in Fig. 3 is 4, excluding the diagonal coefficient, and the height of column 6 is 4, excluding the diagonal coefficient.

The parallel-vector Choleski method, described in Section 3, uses a variable-band storage scheme to achieve optimal vector performance combined with the skyline column heights to avoid calculations with zeros outside the skyline.

3. Parallel-Vector Choleski Method Development

Basic Sequential Choleski Method

In the sequential Choleski method, a symmetric, positive-definite stiffness matrix, [K], can be decomposed as

$$[K] = [U]^T [U] \tag{1}$$

with the coefficients of the upper-triangular matrix, [U]:

$$u_{ij} = 0 \quad \text{for } i > j \tag{2}$$

$$u_{11} = \sqrt{K_{11}}\ ;\ u_{1j} = \frac{K_{1j}}{u_{11}} \quad \text{for } j \geq 1 \tag{3}$$

$$u_{ii} = \sqrt{K_{ii} - \sum_{k=1}^{i-1} u_{ki}^2} \quad \text{for } i > 1 \tag{4}$$

$$u_{ij} = \frac{K_{ij} - \sum_{k=1}^{i-1} u_{ki} u_{kj}}{u_{ii}} \quad \text{for } i, j > 1 \tag{5}$$

When j=i, the numerator of Eq. 5 is identical to Eq. 4 without the square root operation, which simplifies coding.

Regardless of whether the Choleski or Gauss method is used (see Appendix A), the basic skeleton FORTRAN sequential code for matrix factorization is given in Fig. 4 with comments inserted to explain the connection to Eqs. 3-5.

```
         DO 1  i = row#1, row#n
         DO 2  k = top row# of ith column, i-1
c        compute multiplication factor, xmult
         xmult = U(k,i)
cgauss   xmult = U(k,k) * U(k,i) replaces above statement
         DO 3  j = i, k + row length of row k
c        calculate the numerator of Eq. 5
                U(i,j) = K(i,j) - xmult * U(k,j)
   3     Continue
   2     Continue
c        calculate final value of U(i,i) as in Eq. 4
         U(i,i) = SQRT(U(i,i))
cgauss   remove above statement
c        DO loop 4 divides the numerator of Eq. 5 by uii
         xinv = 1/U(i,i)
         DO 4 j = i+1, i + row length of row i
         U(i,j) = U(i,j) * xinv
   4     Continue
   1     Continue
```

Fig. 4 Sequential Choleski variable-band skeleton code for matrix factorization.

To use the Gauss solution method (i.e., for non-positive-definite systems of equations, see Appendix A), only two FORTRAN statements, labeled cgauss in Fig. 4, change.

The multiplier constants, xmult, and the column height information[16,20] are utilized in the DO 2 loop in Fig. 4 to avoid operations with zeros outside the column height (or skyline). The parameter, k, of the DO 2 loop is illustrated in Fig. 3. For i=6 (in DO 1 of Fig. 4), the index k (in DO 2) has the values from 2 to 5 as shown in Fig. 3.

Although [K] and [U] are two-dimensional arrays in Fig. 4, in the actual Choleski factorization code, both are stored in a one-dimensional array (as in Table 3 of 16). The modifications required for the basic, sequential Choleski code to achieve optimal vector and parallel performance (i.e., minimal solution time) are given next.

Vectorize Choleski Code with Loop Unrolling

For a single processor with vector capability, the loop-unrolling technique (suitable for SAXPY operations) can be exploited to significantly improve performance. The SAXPY operation is one of the most efficient computations on vector computers since vector operations are performed in parallel on separate add and multiply functional units.

In Fig. 3, for example, once the first four rows of the factored matrix, [U], have been completely updated, row 5 can be updated according to the numerator of Eq. 5:

$$u_{5j} = k_{5j} - u_{15} * u_{1j} - u_{25} * u_{2j} - u_{35} * u_{3j} - u_{45} * u_{4j} \qquad (6)$$

In Eq. 6, u_{15}, u_{25}, u_{35} and u_{45} are multiplier constants. Thus, u_{15} (or u_{25}, u_{35}, u_{45}), u_{1j} (or u_{2j}, u_{3j}, u_{4j}) and k_{5j} play the role of the terms a, x and y, respectively, in SAXPY operations. The SAXPY operations in Eq. 6 are also loop unrolled to level 4 since operations on four rows are stacked together into one FORTRAN arithmetic statement. This loop unrolling is possible since "partial" updated values of row 5 can be computed when any of the first four rows are complete.

In a previous paper using the column-oriented Choleski method[16], once the first four columns of the factored matrix, [U], were completely updated, all terms of column 5 were updated. For example, u_{25} was computed by Eq. 5 as:

$$u_{25} = \frac{k_{25} - (u_{12} * u_{15})}{u_{22}} \qquad (7)$$

The term u_{25} in Eq. 7 was computed directly as the "final" updated value, and could not be expressed in terms of "partial" updates as is the case in Eq. 6. Therefore, the loop unrolling technique could not be used in this case. Instead, a vector unrolling strategy[16] was used to improve the vector performance in Eq. 5.

However, in the present paper, the sequential Choleski code in Fig. 4 can be modified to include loop-unrolling, say to level 4 as is shown in Fig. 5.

```
         DO 1  i = row#1, row#n
         DO 2  k = top row# of ith column, i-1, 4
         DO 3  j = i, k + row length of row k
c        Eq. 6 (numerator of Eq. 5) code follows
            U(i,j) = K(i,j) - U(k,i) * U(k,j)
                            - U(k+1,i) * U(k+1,j)
                            - U(k+2,i) * U(k+2,j)
                            - U(k+3,i) * U(k+3,j)
   3     Continue
   2     Continue
c        repeat loop 2 to update ith row by extra k values
c        for DO 2 k = 1, 10, 4, extra k values are 9,10
         U(i,i) = SQRT(U(i,i))
         xinv = 1/U(i,i)
         DO 4 j = i+1, i + row length of row i
         U(i,j) = U(i,j) * xinv
   4     Continue
   1     Continue
```

Fig. 5 Vectorized Choleski factorization code (with level 4 loop unrolling).

Using the loop-unrolling technique, the total number of load and store instructions and operations between the main memory and the vector registers is reduced significantly for nested DO-loops. The modified outer loop (DO 2 in Fig. 5), has an increment equal to the level of unrolling, while the innermost loop (DO 3 in Fig. 5) contains more arithmetic computations in a single FORTRAN statement than the basic code. For vector supercomputers, such as

Cray, SAXPY operations are known to be faster than dot-product operations used in the skyline method. The use of a variable-band is preferred to the skyline storage scheme since it permits the SAXPY operations of Eq. 6.

In addition to vector capability, modern high-performance computers also have multiple processors which can operate in parallel. Considerably more work is required by engineers to achieve parallel performance gains than to achieve vector performance gains, since code must be restructured for processor synchronization and load balancing. The parallel-vector Choleski method was coded (in the **Force** parallel FORTRAN language) as the computer program **pvsolve**. **Pvsolve** will be described after a brief synopsis of **Force**.

Parallel FORTRAN Language, Force

Force is a preprocessor which produces executable parallel code from a combination of FORTRAN and a set of simple, yet portable, parallel extensions tailored to run efficiently on parallel computers[17]. The parallel extensions used in **pvsolve** are **Prescheduled** DO, **Shared** and **Private** variables, **Produce** and **Copy**. **Prescheduled** DO causes all processors to execute the same DO-loop statements in parallel simultaneously with each processor using a different DO-loop index. Variables can be either **Shared** between all processors or **Private** (each processor has its own value for the same variable name). Care should be taken to avoid large **Private** arrays, as they are stored in different memory locations for each processor. Therefore, **Shared** arrays are preferred to **Private** arrays. **Copy** and **Produce** are used to synchronize tasks. **Copy** X into Y stores X in Y only if X is "full" (i.e., a signal to all processors to resume their computations), otherwise the processor waits. **Produce** X = K assigns K to X and marks X as "full". If X is "full", **Produce** waits until X is "empty" (i.e., a signal for processors to wait) before assigning K to X. **Force** permits algorithms to be independent of both the computer and the number of processors, as the number of processors is not specified until run time.

Parallel-Vector Choleski Factorization

In Choleski-based methods, a symmetric, positive definite stiffness matrix, [K], can be decomposed as shown in Eq. 1.

For example, u_{57} can be computed from Eq. 5 as:

$$u_{57} = \frac{k_{57}-u_{15}u_{17}-u_{25}u_{27}-u_{35}u_{37}-u_{45}u_{47}}{u_{55}} \tag{8}$$

The calculations in Eq. 8 for the term u_{57} (of row 5) only involve columns 5 and 7. Furthermore, the "final value" of u_{57} cannot be computed until the final, updated values of the first four rows have been completed. Assuming that only the first two rows of the factored matrix, [U], have been completed, one still can compute the second partially-updated value of u_{57} as designated by superscript (2):

$$u_{57}^{(2)} = k_{57} - u_{15}\,u_{17} - u_{25}\,u_{27} \tag{9}$$

If row 3 has also been completely updated, then the third partially-updated value of u_{57} can be calculated as:

$$u_{57}^{(3)} = u_{57}^{(2)} - u_{35}\,u_{37} \tag{10}$$

This observation suggests an efficient way to perform Choleski factorization in parallel on NP processors. For example, each row of the coefficient stiffness matrix, [K], is assigned to a separate processor.

From Equation 8, assuming NP = 4, it is seen that row 5 cannot be completely updated until row 4 has been completely updated. In general, in order to update the i^{th} row, the previous (i-1) rows must already have been updated. For the above reasons, any NP consecutive rows of the coefficient stiffness matrix, [K], will be processed by NP separate processors. As a consequence, while row 5 is being processed by a particular processor, say processor 1, then the first (5-NP) rows have already been completely updated. Thus, if the i^{th} row is being processed by the p^{th} processor, there is no need to check every row (from row 1 to row i-1) to make sure they have been completed. It is safe to assume that the first (i-NP) rows have already been completed as shown in the triangular cross-hatched region of Fig. 6.

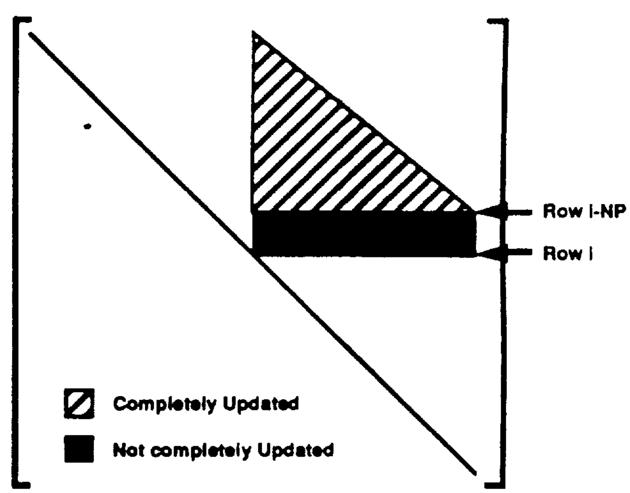


Fig. 6 Information required to update row i.

Synchronization checks are required only for the rows between (i-NP+1) and (i-1) as shown in the rectangular solid region of Fig. 6. Since the first (i-NP) rows have already been completely factored, the i^{th} row can be "partially" processed by the p^{th} processor as shown in Equations 9-10.

The vectorized Choleski code in Fig. 5 has been modified for parallel processing. The resulting skeleton factorization part of the full **pvsolve** code is shown in Fig. 7 with parallel (**Force**) statements in boldface type.

```
        Shared K(21090396)
        Private i,j,k,temp,xinv
c       {X} vector used to indicate when row is finished
        [U] overwrites [K] in actual code to reduce storage
c       calculate U(1,1) in Eq. 3 on one processor
        U(1,1) = SQRT(K(1,1))
c       divide row#1 by U(1,1) as in Eq. 3
c       declare row#1 finished
        Produce X(1) = U(1,1)
c       start all available processors
        Presched DO 1 i = row#2, row#n
c       lock processor if row# (i-NP) is not finished
c       release lock when row is finished
        IF(i-NP.GT. 0) then
        Copy X(i-NP) into temp
        End if
        DO 2 k = top row# of the ith column, i-NP, 4
c       skip DO 3 if all multipliers are zero: zero checking
                DO 3 j = i, k + rowlength of row k
                U(i,j) = K(i,j) - U(k,i) * U(k,j)
                        - U(k+1,i) * U(k+1,j)
                        - U(k+2,i) * U(k+2,j)
                        - U(k+3,i) * U(k+3,j)
  3     continue
  2     continue
c       lock the processor if row# (i-1) not finished
c       release the lock when row#(i-1) is finished
        Copy X(i-1) into temp
        DO 4 k=max(top row# of ith column, i-NP+1), i-1
        DO 5 j = i, k + rowlength of row k
        U(i,j) = U(i,j) - U(k,i) * U(k,j)
  5     continue
  4     continue
        U(i,i) = SQRT(U(i,i))
        xinv = 1/U(i,i)
        DO 6 j = i+1, i + rowlength of row i
        U(i,j) = U(i,j) * xinv
  6     continue
c       broadcast to all processors that row i is finished
        Produce X(i) = U(i,i)
  1     End Presched DO
```

Fig. 7 Parallel-vector Choleski skeleton code (with level 4 loop unrolling).

Solution of Triangular Systems

The forward/backward solution can be made parallel in the outermost loop by using synchronization statements, and can result in excellent computation speed-up for an increasing number of processors on computers where synchronization time is fast compared to computation time. However, on Cray computers, the computations for the forward/backward solution time are so fast that for better performance in **pvsolve**, they are done on one processor with long vectors rather than introducing synchronization overhead on multiple processors. A further time reduction for one processor is obtained by using loop unrolling in the forward elimination and vector unrolling[16] (another form of loop unrolling) in the backward substitution.

4. Evaluation of Method for Structural Analyses

To test the effectiveness of **pvsolve**, described in Section 3, two large-scale structural analyses have been performed on the Cray Y-MP supercomputer at NASA Ames Research Center. These analyses involved calculating the static displacements resulting from initial loadings for finite element models of a high speed research aircraft and the space shuttle solid rocket booster (SRB). The aircraft and SRB models were selected as they were large, available finite-element models of interest to NASA. The Cray Y-MP was selected as it is a high-performance supercomputer with parallel-vector capability. To verify the accuracy of the displacements as calculated from the equilibrium equation (i.e. $[K]\{x\} = \{f\}$), the residual vector,

$$\{R\} = [K]\{x\} - \{f\} \qquad (11)$$

is calculated, and the absolute error norm,

$$e_a = \sqrt{\{R\}^T\{R\}} \qquad (12)$$

and strain energy error norm,

$$e_s = \{x\}^T[K]\{x\} - \{x\}^T\{f\} \qquad (13)$$

are evaluated. If no computer roundoff error occurs, all components in the residual vector, $\{R\}$ are zero. However, performing billions of operations during equation solution introduces roundoff which, for accurate solutions, results in small values for $\{R\}$, e_a and e_s in Eqs. 11-13.

The solution times using **pvsolve** for the SRB application were also obtained on Cray 2 supercomputers at NASA Ames and NASA Langley and compared with solution times for the skyline algorithm in a previous paper[16].

In the following applications, code is inserted in **pvsolve** to calculate the elapsed time and number of operations taken by each processor for equation solution. The Cray timing and performance utilities (**timef, hpm, ja** and **second**) are used to measure the time, operations and speed of the equation solution on each processor. For each problem, the number of Million FLoating point OPerations is divided by the solution time, in Seconds, to determine the overall performance rate of the solver in MFLOPS. The timings obtained are conservative, since they were made with other users on the systems. In every case, times would be less and MFLOP rates more if **pvsolve** were run in a dedicated computer environment.

High Speed Research Aircraft

To evaluate the performance of the parallel-vector Choleski solver, a structural static analysis has been performed on a 16,146 degree-of-freedom finite-element model of a high-speed aircraft concept[21], shown in the upper right of Fig. 8.

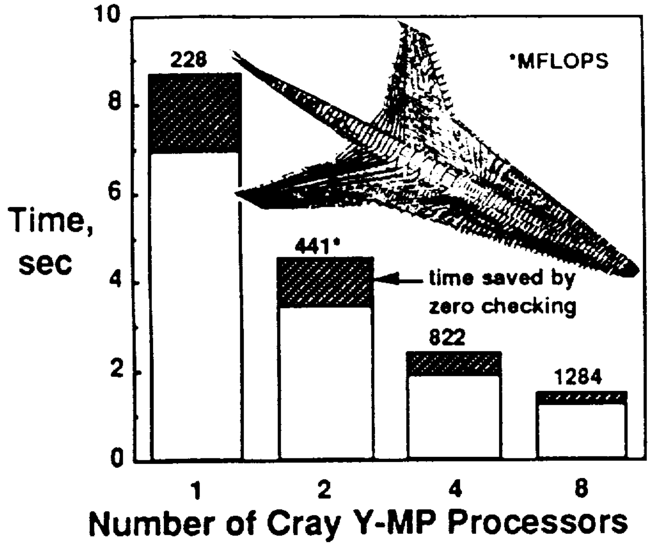


Fig. 8 Effect of more processors on analysis time (High-Speed Research Aircraft).

Since the structure is symmetric, a wing-fuselage half model is used to investigate the overall deflection distribution of the aircraft. The finite element model of the aircraft is generated using the CSM Testbed[18] where the stiffness matrix and load vector are in the form of processor ITER (with reset sipr=-2), described further in Appendix B. The half model contains 2851 nodes, 4329 4-node quadrilateral shell elements, 5189 2-node beam elements and 114 3-node triangular elements. The stiffness matrix for this model has a maximum semi-bandwidth of 600 and an average bandwidth of 321. The half-model is constrained along the plane of the fuselage centerline and subjected to upward loads at the wingtip and the resulting wing and fuselage deflections are calculated.

The numerical accuracy of the static displacements calculated is indicated by the small absolute and strain energy error norms of 0.000009 and 0.000005, respectively, computed from Eqs. 12-13. These residuals are identical no matter how many processors are used. The small values of the residuals indicates that the solution satisfies the original force-displacement equation. The residuals are independant of the number of processors indicating no error is introduced by synchronizing the calculations on multiple processors.

The time taken for a typical finite element code to generate the mesh, form and factor the stiffness matrix is 134 seconds on a Cray Y-MP (802 seconds on a Convex 220) of which the matrix factorization is 51 seconds. Using **pvsolve**, the factorization for this aircraft application requires 2 billion operations which reduces to 1.4 billion when operations with zeros are eliminated. Although CPU time is less for one processor, elapsed time is reported as it is the only meaningful measure of parallel performance. Factoring [K] with no zero checking takes 8.68 and 1.54 elapsed seconds (at a rate of 228 and 1284 MFLOPS) on one and eight Cray Y-MP processors, respectively, as shown in Table 1.

Table 1 Matrix decomposition time (MFLOPS) for aircraft on Cray Y-MP:
16,146 equations, bandwidth=600 max, 321 average
5,579,839 matrix size, 499,505 nonzeros

Processors	Sec (MFLOPS)	Sec (MFLOPS) with zero-checking
1	8.68 (228)	6.81 (203)
2	4.50 (441)	3.46 (399)
4	2.41 (822)	1.89 (730)
8	1.54 (1284)	1.29 (1071)

Eliminating operations with zeros within the variable bandwidth (zero checking, see Fig. 7) further reduces the solution time to 6.81 and 1.29 seconds, respectively, on one and eight processors. However, the reduced time with zero checking is accompanied by a reduction in computation rate (MFLOPS), since the added IF statements also reduce the number of operations. The reduction in computation time (nearly proportional to the number of processors) and the portion of time saved by zero-checking are shown in Fig. 8. The number above the bars (in MFLOPS) in Fig. 8 show the increased computation rate as the number of processors increases.

Space Shuttle Solid Rocket Booster (SRB)

In addition to the high-speed aircraft, the static displacements of a two-dimensional shell model of the space shuttle SRB, shown in the upper right of Fig. 9, have been calculated.

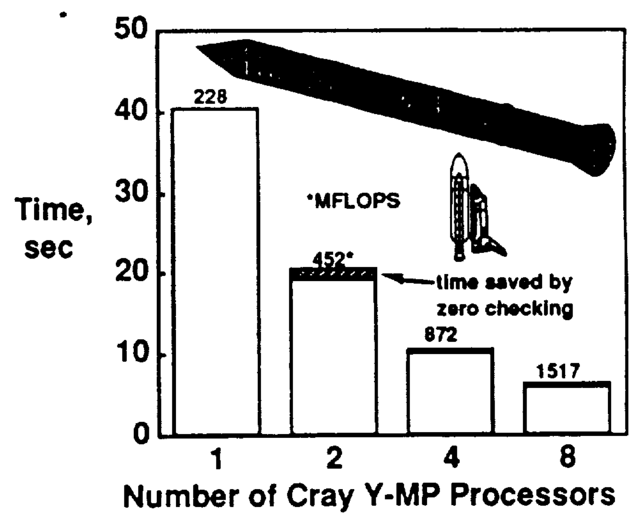


Fig. 9 Effect of more processors on analysis time (Space Shuttle SRB).

This SRB model is used to investigate the overall deflection distribution for the SRB when subjected to mechanical loads corresponding to selected times during the launch sequence[22]. The model contains 9205 nodes, 9156 4-node quadrilateral shell elements, 1273 2-node beam elements and 90 3-node triangular elements, with a total of 54,870 degrees of freedom. The stiffness matrix for this application has a maximum semi-bandwidth of 900 and an average bandwidth of 383. A detailed description and analysis of this problem is given in references **22** and **23**.

The calculated absolute and strain energy residuals for the static displacements are 0.00014 and 0.0017, respectively, from Eqs. 12-13. This accuracy indicates that roundoff error in the displacement calculations is insignificant despite the 9.2 billion floating point operations performed.

The time for a typical finite element code to generate the mesh, form and factor the stiffness matrix is 391 seconds on the Cray Y-MP (15 hours on a VAX 11/785) of which the matrix factorization is 233 seconds (51,185 seconds on VAX). Using **pvsolve**, the factorization for this SRB problem, requires 40.26 and 6.04 seconds on one and eight Cray Y-MP processors, respectively, as shown in Table 2. Eliminating more than one billion operations on zeros further reduces the solution time to 5.79 seconds on eight processors but reduces the computation rate to 1444 MFLOPS. The CPU times are approximately 10 percent less than the elapsed times quoted on one processor.

Table 2 Matrix decomposition time (MFLOPS) (shuttle SRB on Cray Y-MP)

54,870 equations,bandwidth=900 max, 383 average
21,090,396 matrix size, 1,310,973 nonzeros

Processors	Sec. (MFLOPS)	Sec. (MFLOPS) with zero-checking
1	40.26 (228)	40.97 (224)
2	20.27 (452)	19.32 (425)
4	10.50 (872)	10.00 (821)
8	6.04 (1517)	5.79 (1444)

A reduction in matrix decomposition time by a factor of 7.08 on eight processors compared to one processor (for zero checking) is shown in Fig. 9. The corresponding computation rate for this matrix factorization, using eight processors on the Cray Y-MP is 1,517 MFLOPS. The previous fastest time to solve this problem on the Cray Y-MP using a sparse solver was 23 seconds on one processor and 9 seconds on eight processors for a speedup factor of 2.5[7,24].

For structural analysis problems with a larger average column height, and bandwidth than the aircraft or SRB discussed, one can expect **pvsolve** to perform computations at even higher MFLOPS rates since the majority of the vector operations are performed on long vectors. For example, a rate of 1784 MFLOPS has been achieved by **pvsolve** for a structural matrix with an average bandwidth of 699 on the eight-processor Cray Y-MP[25-26].

The decomposition time for the Shuttle SRB matrix using **pvsolve**, is compared to the skyline algorithm[16] in Fig. 10 for 1, 2 and 4 Cray 2 processors.

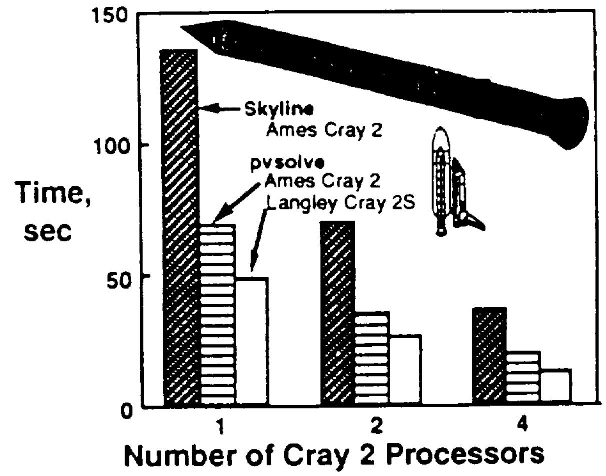


Fig. 10 SRB decomposition time comparison (pvsolve vs. skyline method[16]).

A reduction in decomposition time by a factor of 2 is shown for **pvsolve** in the figure for the Cray 2 at NASA Ames. An additional reduction in decomposition time of approximately 50 percent is shown for **pvsolve** on the newer Cray 2S at NASA Langley with faster memory access using static RAM compared to dynamic RAM on the Cray 2 at NASA Ames. The decomposition time for **pvsolve** using eight processors on the Cray Y-MP (six seconds in Fig. 9) is a reduction by factors of 23 and 6 when compared to the skyline solution on 1 and 4 Cray 2 processors, respectively, shown in Fig. 10.

The above results have been obtained using loop unrolling to level 9. On the Cray Y-MP supercomputer, the performance continues to increase until loop unrolling level 9, after which further performance gains are not significant compared to the complex coding required. The **pvsolve** code performed best with an odd number for loop unrolling, because both data paths to memory are used simultaneously at all times. The vector being modified plus the 9 unrolling vectors make ten total vectors, an even number, which keeps both data paths busy.

5. Concluding Remarks

A parallel-vector Choleski method for the solution of large-scale structural analysis problems has been developed and tested on Cray supercomputers. The method exploits both the parallel and vector capabilities of modern high-performance computers. To minimize computation time, the method performs parallel computation at the outermost DO-loop of the matrix factorization, the most time-consuming part of the equation solution. In addition, the most intensive computations of the factorization, the innermost DO-loop has been vectorized using a SAXPY-based scheme. This scheme allows the use of the loop-unrolling technique which minimizes computation time. The forward and backward solution phases have been found to be more effective to perform sequentially with loop-unrolling and vector-unrolling, respectively.

The parallel-vector Choleski method has been used to calculate the static displacements for two large-scale structural analysis problems; a high-speed aircraft and the space shuttle solid rocket booster. For both structural analyses, the static displacements are calculated with a high degree of accuracy as indicated by the small values of the absolute and strain energy error norms. The total equation solution time is small for one processor and is further reduced in proportion to the number of processors. The option to avoid operations with internal zeros in the matrix further reduces both the number of operations and the computation time for both applications.

Factoring the stiffness matrix for the space shuttle solid rocket booster, which formerly required hours on most computers and minutes on supercomputers by other methods, has been reduced to seconds using the parallel-vector variable-band Choleski method. The speed of **pvsolve** should give engineers and designers the opportunity to include more design variables and constraints during structural optimization and to use more refined finite-element meshes to obtain an improved understanding of the complex behavior of aerospace structures leading to better, safer designs. Since the algorithm is independent of the number of processors, it is not only attractive for current supercomputers, but also for the next generation of shared-memory supercomputers, where the number of processors is expected to increase significantly.

6. Appendix A

The row-oriented, sequential versions of both the Choleski and Gauss methods are presented together to illustrate how their basic operations are closely related and readily identified. To simplify the discussion, the following system of equations is used throughout this section:

$$[K]\{x\} = \{f\} \tag{14}$$

where
$$[K] = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix} \tag{15}$$

and
$$\{f\} = \begin{Bmatrix} 1 \\ 0 \\ 0 \end{Bmatrix} \tag{16}$$

The solution of equations 14-16 is:

$$\{x\} = \begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} \tag{17}$$

The basic idea in both the Choleski and Gauss elimination methods is to reduce the given coefficient matrix, [K], to an upper triangular matrix, [U]. This process can be accomplished with appropriate row operations. The unknown vector, {x}, can be solved by the familiar forward and backward substitution.

Choleski Method

The stiffness matrix [K] of equation 15 can be converted into a Choleski upper-triangular matrix, [U], by appropriate row operations:

$$[K1] = [K] = \begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\Rightarrow [K2] = \begin{bmatrix} \sqrt{2} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & -1 & 1 \end{bmatrix} \Rightarrow [K3] = \begin{bmatrix} \sqrt{2} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}}{\sqrt{2}} & -\frac{\sqrt{2}}{\sqrt{3}} \\ 0 & -1 & 1 \end{bmatrix}$$

$$\Rightarrow [K4] = \begin{bmatrix} \sqrt{2} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}}{\sqrt{2}} & -\frac{\sqrt{2}}{\sqrt{3}} \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \Rightarrow [K5] = \begin{bmatrix} \sqrt{2} & \frac{-1}{\sqrt{2}} & 0 \\ 0 & \frac{\sqrt{3}}{\sqrt{2}} & -\frac{\sqrt{2}}{\sqrt{3}} \\ 0 & 0 & \frac{1}{\sqrt{3}} \end{bmatrix}$$

where

Row 1 of [K2] = Row 1 of [K] $/\sqrt{K1(1,1)}$
Row 2 of [K2] = Row 1 of [K2] $/\sqrt{2}$ + Row 2 of [K1]
Row 2 of [K3] = Row 2 of [K2] $/\sqrt{K2(2,2)}$
Row 3 of [K4] = Row 2 of [K3] $* \sqrt{\frac{2}{3}}$ + Row 3 of [K3]
Row 3 of [K5] = Row 3 of [K4] $/\sqrt{K4(3,3)}$

The multiplier constants, m_{ij}, used in the forward substitution (or updating the right-hand side vector of Eq. 14) are the same as terms in the factorized upper-triangular matrix such that:

$$m_{12} = u_{12} = -\frac{1}{\sqrt{2}}, \; m_{13} = u_{13} = 0, \; m_{23} = u_{23} = -\frac{\sqrt{2}}{\sqrt{3}}$$

Gauss Elimination Method

As in the Choleski Method just described, the stiffness matrix, [K], of Eq. 15 can be converted into a Gauss upper-triangular matrix by appropriate row operations.

$$[K1]=[K]=\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\Rightarrow [K2]=\begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & -1 & 1 \end{bmatrix} \Rightarrow [K3]=\begin{bmatrix} 2 & -1 & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & 0 & \frac{1}{3} \end{bmatrix}$$

In this version of Gauss elimination, the multipliers m_{ij} can be obtained from the factored matrix, [U], as:

$$m_{12}=\frac{u_{12}}{u_{11}}=-\frac{1}{2}$$
$$m_{13}=\frac{u_{13}}{u_{11}}=\frac{0}{2}=0$$
$$m_{23}=\frac{u_{23}}{u_{22}}=\frac{-1}{\frac{3}{2}}=-\frac{2}{3}$$

An alternative version of Gauss elimination where the final diagonal elements become 1 follows:

$$[K1]=[K]=\begin{bmatrix} 2 & -1 & 0 \\ -1 & 2 & -1 \\ 0 & -1 & 1 \end{bmatrix}$$

$$\Rightarrow [K2]=\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & \frac{3}{2} & -1 \\ 0 & -1 & 1 \end{bmatrix} \Rightarrow [K3]=\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & -\frac{2}{3} \\ 0 & -1 & 1 \end{bmatrix}$$

$$\Rightarrow [K4]=\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & -\frac{2}{3} \\ 0 & 0 & \frac{1}{3} \end{bmatrix} \Rightarrow [K5]=\begin{bmatrix} 1 & -\frac{1}{2} & 0 \\ 0 & 1 & -\frac{2}{3} \\ 0 & 0 & 1 \end{bmatrix}$$

Since the final diagonal terms become one, in the computer code, the main diagonal of the factored matrix is used to store the diagonal terms before scaling.

For example, $u_{11}=2$; $u_{22}=\frac{3}{2}$; and $u_{33}=\frac{1}{3}$. The multiplier m_{ij} is obtained from the factored matrix, [U], as:

$$m_{12}=u_{12} * u_{11}=-\frac{1}{2}\times 2=-1$$
$$m_{13}=u_{13} * u_{11}=0\times 2=0$$
$$m_{23}=u_{23} * u_{22}=-\frac{2}{3}\times\frac{3}{2}=-1$$

Similarities of Choleski and Gauss Method

1) The Choleski and Gauss solution procedures are quite similar since both methods can be expressed in terms of row operations which differ only by the scale-factors as explained above.

2) For both methods, the multipliers, m_{ij}, used in the forward substitution (to update the right-hand-side vector of Eq. 14) can always be recovered conveniently from the factored, upper triangular matrix, [U].

3) Both methods can be adapted to solve unsymmetric systems of linear equations. The basic procedure is essentially the same as that outlined above except that the computer storage increases since the lower triangle matrix of the factored matrix is used to store the multipliers, m_{ij}. In some applications, partial pivoting may be useful.

4) Since the multipliers of the Choleski method are identical to its factored, upper triangular matrix, [U], the Choleski method is slightly more efficient than the Gauss method However, the Gauss method can also be used to solve non-positive-definite systems of equations.

7. Appendix B

The input data and arguments required to call the equation solver, **pvsolve**, together with a simple 21-equation example are given in this Appendix. The user should have a limited knowledge of parallel computing and the parallel FORTRAN language **Force**[17]. **Pvsolve** contains a **Force** subroutine, PVS, which may be called by general purpose codes. The information required by PVS to solve systems of simultaneous equations (i.e., [K]{u} = {f}) is transferred via arguments in the call statement:

```
Forcecall PVS(a,b,maxa,irowl,icolh,neq,nterms,iif,opf)
```

where:

a = a real vector, dimensioned nterms, containing the coefficients of the stiffness matrix, [K].

b = a real vector, dimensioned neq, containing the load vector, {f}. Upon return from subroutine PVS, b contains the displacement solution, {u}.

maxa = an integer vector, dimensioned neq, containing the location of the diagonal terms of [K] in vector {a}, equal to the sum of the number coefficients.

irowl = an integer vector, dimensioned neq, containing the row lengths (i.e., half-bandwidth of each row excluding the diagonal term) of [K].

icolh = an integer vector, dimensioned neq, containing the column heights (excluding the diagonal term) of each column of the stiffness matrix, [K].

neq = number of equations to solve (= degrees of freedom).

nterms = the dimension of the vector, {a}, [= maxa(neq)].

iif = 1 factor system of equations without internal zero check
= 2 factor system of equations with internal zero check
= 4 perform forward/backward substitution
= 5 perform forward/backward substitution and error check

opf, ops = an integer vector, dimensioned to the number of processors (8 for Cray Y-MP), containing the number of operations performed by each processor during factor and solve, respectively.

For example, the values of these input variables to solve a system of 21 equations, whose right hand side is the vector of real numbers from 1. to 21., and [K] is the symmetric, positive-definite matrix in Fig. B1 are given in Table B1.

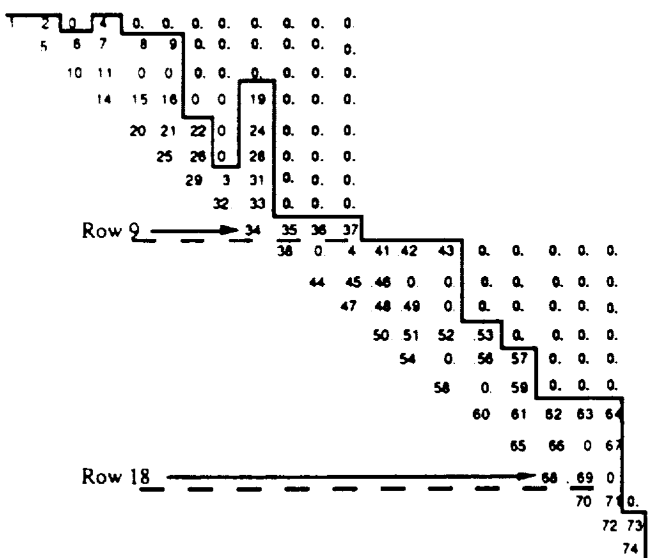


Fig. B1 Example [K] matrix with 21 equations.

The line in Fig. B1 represents the skyline defined by the column heights which extend up to the last nonzero in each column. The "extra zeros" outside the skyline (in boldface in Fig. B1) are required to achieve level 9 loop unrolling. The DO 2 loop in Fig. 5 illustrates this for level 4 loop unrolling. The vectors {a}, {b}, {maxa}, {icolh}, and {irowl} which are read by pvsolve are given in Table B1:

Table B1 Pvsolve input to solve [K]{x}={b} (example with 21 equations)

i	a(i)	b(i)	maxa(i)	icolh(i)	irowl(i)
1	1.	1.	1	0	11
2	.2	2.	13	1	10
3	0	3.	24	1	9
4	.4	4.	34	3	8
5	0	5.	43	3	7
6	0	6.	51	4	6
7	0	7.	58	2	5
8	0	8.	64	1	4
9	0	9.	69	5	3
10	0	10.	73	1	10
11	0	11.	84	2	9
12	0	12.	94	3	8
13	5.	13.	103	3	7
14	.6	14.	111	4	6
15	.7	15.	118	5	5
16	.8	16.	124	3	4
17	.9	17.	129	3	3
18	0	18.	133	2	2
19	0	19.	136	3	2
20	0	20.	139	4	1
21	0	21.	141	1	0
22	0				
23	0				
24	10.				
25	.11				
26-33	0				
34	14.				
35	.15				
36	.16				
37-38*	0				
39	.19				
.	.				
.	.				
.	.				
135	0				
136	70.				
137	.71				
138	0				
139	72.				
140	.73				
141	74.				

where neq = 21 and nterms = 141. This input data is read at the beginning of the **pvsolve** program from the file 'COEFS.COLM' by subroutine CSMIN (see listing in Appendix C). The Force subroutine, PVS is then called twice; first to factor the matrix (iif = 2), and second to perform the forward/backward solution for displacements with error checking (iif = 5). A record is kept of number of floating point operations performed by each processor to factor and solve the matrix (totf, tots) as well as the elapsed (et0-et5) and task CPU time (t0-t5) on each processor at six key stages in the solution. Subroutine NORM reads the original matrix and load vector from the file 'COEFS.COLM' and evaluates the residual (Eq. 11) and the error norms (Eqs. 12-13).

8. Appendix C

A listing of the parallel-vector solution algorithm, **pvsolve**, coded in the parallel FORTRAN language, **Force**[17], follows in this Appendix. The code extends the skeleton code in Fig. 7 considerably by using loops unrolled to level 9 (instead of 4), one-dimensional vectors with pointers (instead of arrays) and by including the code for input/output, data handling, initialization, timing and counting operations. Following the **pvsolve** code is the command file used to obtain the static displacements for the aircraft and SRB structures using the Solid State Disk and 1,2,4 and 8 Cray Y-MP processors. The **pvsolve** code is all FORTRAN except for the **cdir$ ivdep** vector directive, and the **Force** parallel directives in **boldface type**. The dimension of the variables given on line 2 is for the static analysis of the 16,146 equation research aircraft and should be replaced by the dimensions given in line 3 to obtain the space shuttle SRB displacement solution All variables are **Private** unless they are declared as **Shared**.

```
      Force PVSOLVE of np ident me
      Shared real a(5208900),b(16150),at(499600),opf(8)
csrb  Shared real a(21090500),b(54890),at(1350761)
      Shared real t0(8),t1(8),t2(8),t3(8),t4(8),t5(8),ops(8)
      Shared real et0(8),et1(8),et2(8),et3(8),et4(8),et5(8)
      Shared integer maxa(16150),irow(16150),irowl(16150)
      Shared integer icoln(499600),icolh(16150),nc,neq
      End declarations
         et0(me)=timef()/1000.
         t0(me)=second()/np
       if (me.eq.1) then call CSMIN(a,b,maxa,irowl,icolh,neq,
     +    nterms,irow,icoln,nc,maxbw,8,locrow,iavebw)
        write(*,*)'* PVSOLVE - pvsolve - PVSOLVE Mar. 1990'
        write(*,*)'* Parallel-Vector equation SOLVEr by Olaf'
        write(*,*)'* Storaasli, Tarun Agarwal and Duc Nguyen'
        write(*,*)'* ',np,' proc. solve ',neq,' equations, nc= ',nc
        write(*,*)'* bandwidth: max= ',maxbw,', avg.= ',iavebw
        write(*,*)'* [k] matrix size, nterms= ',nterms,' words'
        endif
         et1(me)=timef()/1000.
         t1(me)=second()/np
      Barrier
      End barrier
         et2(me)=timef()/1000.
         t2(me)=second()/np
 call PVS to factor [k] with internal zero check (iif = 2)........
         iif = 2
      Forcecall PVS(a,b,maxa,irowl,icolh,neq,nterms,iif,opf(me))
         et3(me)=timef()/1000.
         t3(me)=second()/np
 call PVS to backsolve for {u} (iif = 4, 5 error check eqs. 11-13)
         iif = 5
      Forcecall PVS(a,b,maxa,irowl,icolh,neq,nterms,iif,ops(me))
         et4(me)=timef()/1000.
         t4(me)=second()/np
       Barrier
         nat=499600
         umax = abs(b(1))
         do 1 i=1,neq
 1       umax = amax1(umax,abs(b(i)))
        write(*,*)'* Maximum displacement = ',umax
        if(iif.eq.5 ) call NORM(irowl,icoln,b,neq,nc)
c......reorder displacements and write to CSM Testbed.........
         call TOCSM(b,irowl,icoln,at,at,icoln,8,nat)
          tmax1=0
          tmax2=0
          tmax3=0
          totf=0
          tots=0
        write(*,*)'** elapsed & cpu task time (sec) *****'
        write(*,*)'proc.  force  input  Barrier  factor  f/b'
        do 2 i=1,np
         write(*,3)'wall ',i,et0(i),et1(i),et2(i),et3(i),et4(i)
         write(*,3)'tcpu ',i,t0(i),t1(i),t2(i),t3(i),t4(i)
          tmax1=max(tmax1,et3(i)-et2(i))
          tmax2=max(tmax2,et4(i)-et3(i))
          tmax3=max(tmax3,et4(i)-et2(i))
          totf=totf+opf(i)/1000000.
 2        tots=tots+ops(i)/1000000.
 3       format(a,i2,5f9.5)
        write(*,*) tmax1,' secs decomp, ',totf,
     + ' million ops. at ',totf/tmax1,' mflops '
         write(*,*) tmax2,' secs solve , ',tots,
     + ' million ops. at ',tots/tmax2,' mflops'
        write(*,*) tmax3,' secs TOTAL , ',totf+tots,
     + ' million ops. at ',(tots+totf)/tmax3,' mflops'
      End barrier
         et5(me)=timef()/1000.
         t5(me)=second()/np
        write(*,*)'proc. ',me,' tot wall=',et5(me),'tcpu=',t5(me)
         call exit(0)
      Join
       end
      Forcesub PVS(a,b,maxa,irowl,icolh,neq,nterms,iif,ops)
     +    of np ident me
      dimension a(*),b(*),icolh(*),maxa(*),irowl(*)
      Async real x(16150)
      End declarations
       if(iif.le.2) then
      Presched do 9 i = 1, neq
      Void x(i)
 9    End presched do
         ops = 0
      Barrier
         a(1) = sqrt(a(1))
         xinv= 1.0/a(1)
cdir$ ivdep
         do 20 k = 1, irowl(1)
 20        a(k+1) = xinv*a(k+1)
         ops = ops + irowl(1)+2
      Produce x(1)=a(1)
      End barrier
c.....factor stiffness matrix in parallel from row 2 to neq
      Presched do 100 i = 2, neq
           im1 = maxa(i)
           ic1 = icolh(i)
c......get indices to segment column i in 3 parts.............
         ibot = i - 9*( (i-1)/9 )
         icol = ic1 - ibot + 1
         icolp= icol/9
         itop = icol - 9*icolp
         jrow = i - ic1
         jm1  = maxa(jrow) + ic1
         jjrow=irowl(jrow)
         if (itop. ge. 1) then
         icopy = jrow + itop - 1
         if (isfull(x(icopy))) go to 331
       Copy x(icopy) into temp
         endif
c...........................................................................
 331     go to (101,102,103,104,105,106,107,108), itop
```

```
      go to 150
cdir$ ivdep
101    do 111 k = 1, jjrow-ic1+1
              km1 = k-1
111      a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
      go to 150
102           jm2 = jm1 + jjrow
cdir$ ivdep
       do 112 k = 1, jjrow-ic1+1
              km1 = k-1
112       a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +                           -a(jm2)*a(jm2+km1)
      go to 150
103           jm2 = jm1 + jjrow
              jm3 = jm2 + jjrow -1
cdir$ ivdep
       do 113 k = 1, jjrow -ic1+1
              km1 = k -1
113         a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
      go to 150
104           jm2 = jm1 + jjrow
              jm3 = jm2 + jjrow -1
              jm4 = jm3 + jjrow -2
cdir$ ivdep
       do 114 k = 1, jjrow -ic1+1
              km1 = k -1
114         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)
      go to 150
105           jm2 = jm1 + jjrow
              jm3 = jm2 + jjrow -1
              jm4 = jm3 + jjrow -2
              jm5 = jm4 + jjrow -3
cdir$ ivdep
       do 115 k = 1, jjrow -ic1+1
              km1 = k -1
115         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
      go to 150
106           jm2 = jm1 + jjrow
              jm3 = jm2 + jjrow -1
              jm4 = jm3 + jjrow -2
              jm5 = jm4 + jjrow -3
              jm6 = jm5 + jjrow -4
cdir$ ivdep
       do 116 k = 1, jjrow -ic1+1
               km1= k -1
116         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)
      go to 150
107           jm2 = jm1 + jjrow
              jm3 = jm2 + jjrow -1
              jm4 = jm3 + jjrow -2
              jm5 = jm4 + jjrow -3
              jm6 = jm5 + jjrow -4
              jm7 = jm6 + jjrow -5
cdir$ ivdep
       do 117 k = 1, jjrow -ic1+1
              km1 = k -1
117         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
      go to 150
108           jm2 = jm1 + jjrow
              jm3 = jm2 + jjrow -1
              jm4 = jm3 + jjrow -2
              jm5 = jm4 + jjrow -3
              jm6 = jm5 + jjrow -4
              jm7 = jm6 + jjrow -5
              jm8 = jm7 + jjrow -6
cdir$ ivdep
       do 118 k = 1, jjrow -ic1+1
              km1 = k -1
118         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
     +            -a(jm8)*a(jm8+km1)
150   ops = ops + itop*(jjrow -ic1+2)*2
     ll = 1
       idiv = 1
      if (icolp.le.ll) then
       ll =icolp
       idiv1=1
      else
       idiv1=icolp-ll+1
      endif
       jtop = ic1
       jbot = ic1-itop+1
      do 10 l = 1, ll
       jtop = jtop - itop
       jbot = jbot - 9*idiv1
       itop = 9*idiv1
       idiv1 = idiv
      if (l.eq.ll) then
       icopy = i - 1
      else
       icopy = i -jbot +ibot-1
      endif
       if(isfull(x(icopy))) go to 332
   Copy x(icopy) into temp
c.....unroll to level 9: fast vector saxpy operations......
332       do 200 j = jtop, jbot, -9
            jj1 = i-j
            jjrow = irowl(jj1)
            jm1 = maxa(jj1) + j
            jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
            jm6 = jm5 + jjrow -4
            jm7 = jm6 + jjrow -5
            jm8 = jm7 + jjrow -6
            jm9 = jm8 + jjrow -7
       if(iif.eq.2) then
       if (a(jm9).ne.0.0) then
cdir$ ivdep
       do 300 k = 1, irowl(jj1) -j+1
              km1 = k -1
300         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
     +            -a(jm8)*a(jm8+km1)-a(jm9)*a(jm9+km1)
       ops = ops + 18*(irowl(jj1)-j+1)
      else
         if(a(jm4).ne.0.0) then
         go to 301
```

```
        else
          if((a(jm1).eq.0.0).and.(a(jm2).eq.0.0).and.
     +        (a(jm3).eq.0.0)) go to 302
        endif
cdir$  ivdep
301     do 310 k = 1, irowl(jj1) -j +1
              km1 = k -1
310         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)
        ops = ops + 8*(irowl(jj1)-j+1)
302     if((a(jm5).eq.0.0).and.(a(jm6).eq.0.0).and.
     .      (a(jm7).eq.0.0).and.(a(jm8).eq.0.0)) go to 200
cdir$  ivdep
        do 320 k = 1, irowl(jj1) -j +1
              km1 = k -1
320         a(im1+km1) = a(im1+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
     +            -a(jm8)*a(jm8+km1)
        ops = ops + 8*(irowl(jj1)-j+1)
        endif
        else
cdir$  ivdep
        do 330 k = 1, irowl(jj1) - j +1
              km1 = k -1
330         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
     +            -a(jm8)*a(jm8+km1)-a(jm9)*a(jm9+km1)
        ops = ops + 18*(irowl(jj1)-j+1)
       endif
200     continue
10      continue
        ll=i-1
        if (isfull(x(ll))) go to 333
    Copy x(ll) into temp
c.......................................................................
333        go to (201,202,203,204,205,206,207,208) ibot-1
         go to 250
201             jjrow = irowl(i-1)
                jm1 = maxa(i-1) +1
cdir$  ivdep
          do 211 k= 1, jjrow
                km1 = k-1
211             a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1
+km1)
         go to 250
202             jjrow = irowl(i-2)
                jm1 = maxa(i-2) +2
                jm2 = jm1 + jjrow
cdir$  ivdep
            do 212 k = 1, jjrow -1
                        km1 = k -1
212           a(im1+km1)=a(im1+km1)-a(jm1)*a(jm1+km1)
     +                            -a(jm2)*a(jm2+km1)
         go to 250
203             jjrow = irowl(i-3)
                jm1 = maxa(i-3) + 3
                jm2 = jm1 + jjrow
                jm3 = jm2 + jjrow -1
cdir$  ivdep
            do 213 k = 1, jjrow -2
                        km1 = k - 1
213           a(im1+km1)=a(im1+km1)-a(jm1)*a(jm1+km1)
     +           -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
         go to 250
204             jjrow = irowl(i-4)
                jm1 = maxa(i-4) + 4
                jm2 = jm1 + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
cdir$  ivdep
          do 214 k = 1,jjrow  -3
              km1 = k -1
214         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)
        go to 250
205             jjrow = irowl(i-5)
                jm1 = maxa(i-5) + 5
                jm2 = jm1 + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
                jm5 = jm4 + jjrow -3
cdir$  ivdep
          do 215 k = 1, jjrow -4
              km1 = k -1
215         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
         go to 250
206             jjrow = irowl(i-6)
                jm1 = maxa(i-6) +6
                jm2 = jm1 + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
                jm5 = jm4 + jjrow -3
                jm6 = jm5 + jjrow -4
cdir$  ivdep
          do 216 k = 1, jjrow -5
              km1 = k -1
216         a(im1+km1) = a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)
         go to 250
207             jjrow = irowl(i-7)
                jm1 = maxa(i-7)+7
                jm2 = jm1 + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
                jm5 = jm4 + jjrow -3
                jm6 = jm5 + jjrow -4
                jm7 = jm6 + jjrow -5
cdir$  ivdep
          do 217 k = 1, jjrow -6
              km1 = k -1
217         a(im1+km1)=a(im1+km1)-a(jm1)*a(jm1+km1)
     +            -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +            -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +            -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
         go to 250
208             jjrow =irowl(i-8)
                jm1 = maxa(i-8) + 8
                jm2 = jm1 + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
                jm5 = jm4 + jjrow -3
                jm6 = jm5 + jjrow -4
                jm7 = jm6 + jjrow -5
                jm8 = jm7 + jjrow -6
cdir$  ivdep
          do 218 k = 1, jjrow -7
```

```
                 km1 = k -1
218              a(im1+km1)=a(im1+km1)- a(jm1)*a(jm1+km1)
     +                    -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     +                    -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     +                    -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
     +                    -a(jm8)*a(jm8+km1)
250        ops = ops + 2*(ibot-1)*(jjrow -ibot +2)
           a(im1) =sqrt(a(im1))
           xinv = 1.0/a(im1)
cdir$  ivdep
           do 260  k = 1, irowl(i)
260        a(im1+k) = xinv *a(im1+k)
           ops = ops + irowl(i) +2
       Produce x(i) = a(im1)
100  End  presched do
        els e
c.....forward reduction- unroll to level 3 for fast vector speed:
c.....each 3 rows of [k] must end in the same column number..
       Barrier
           ops = 0
           ibot = neq -3* (neq/3)
        do 510 i = 1,neq-ibot,3
          im1 = maxa(i)
          im2 = maxa(i+1)
          im3 = maxa(i+2)
          xmult1 = b(i)/a(im1)
          xmult2 = (b(i+1) - xmult1*a(im1+1))/a(im2)
          xmult3 = (b(i+2) - xmult1*a(im1+2)
     +                       - xmult2*a(im2+1))/a(im3)
          b(i) =   xmult1
          b(i+1) = xmult2
          b(i+2) = xmult3
cdir$  ivdep
           do 520 j = i+3, i+irowl(i)
520           b(j) = b(j) - xmult1*a(im1+j-i)
     +                    - xmult2*a(im2+j-i-1)
     +                    - xmult3*a(im3+j-i-2)
510        ops = ops + 6*(irowl(i)-2)+ 9
          if ( ibot.eq.1) then
           b(neq) = b(neq)/a(maxa(neq))
           ops = ops + 1
         else
          if (ibot.eq.2) then
           im1 = neq -1
           b(im1) =  b(im1)/a(maxa(im1))
           b(neq) = (b(neq) -b(im1)*
     + a(maxa(im1)+1))/a(maxa(neq))
           ops = ops + 4
         endif
         endif
c........back substitution with vector unrolling follows...
           b(neq) = b(neq)/a(maxa(neq))
           ops = ops +1
           jm1 = neq -1
       if (ibot .eq. 2) then
           im1 = neq -1
           b(im1)=(b(im1)-
 a(maxa(im1)+1)*b(neq))/a(maxa(im1))
           ops = ops + 3
           jm1 = neq -2
       endif
       if (ibot .eq. 0) then
           im1 = neq -1
           b(im1)=(b(im1)-a(maxa(im1)+1)*b(neq))/a(maxa(im1))
           im2 = neq -2
           b(im2) =(b(im2)-a(maxa(im2)+1)*b(im1)
     +        -a(maxa(im2)+2)*b(neq))/a(maxa(im2))
           ops = ops + 8
           jm1 = neq -3
       endif
        do 1010 i = jm1,1,-3
           im1 = maxa(i)
           im2 = maxa(i-1)
           im3 = maxa(i-2)
           xmult1 = 0.0
           xmult2 = 0.0
           xmult3 = 0.0
cdir$  ivdep
           do 1020 j=i+1, irowl(i)+i
           xmult1 = xmult1 + a(im1+j-i)*b(j)
           xmult2 = xmult2 + a(im2+j-i+1)*b(j)
1020       xmult3 = xmult3 + a(im3+j-i+2)*b(j)
            b(i) = (b(i) - xmult1)/a(im1)
            b(i-1) = (b(i-1) - a(im2+1)*b(i) - xmult2)/a(im2)
            b(i-2) = (b(i-2)-a(im3+2)*b(i)-a(im3+1)*b(i-1)
     +              -xmult3)/a(im3)
1010        ops = ops + 6*(irowl(i)) +12
         End  barrier
           endif
           return
           end
       subroutine NORM(irow,icoln,x,neq,nc)
        dimension irow(*),icoln(*),x(*),b(neq),diag(neq),offdia(nc
c.....get error error norm: {a}*{x}={b}: read file COEFS.COLM
c..... ([xqt iter with reset sipr=-2 in CSM Testbed) where:
c.....nc=number of nonzero, off-diagonal terms of [k]
c.....irow(neq)=no. of nonzeros in each row w/o diagonal
c.....icoln(nc)=column no. of nonzero terms of [k] by row
c.....diag(neq)=diagonal terms of [k], b(neq)=load vector
c.....offdia(nc)=nonzero, offdiagonal terms of [k]
           rewind(8)
           read(8) neq,neq2,nc,nc2,jdof,jt,ndof
           read(8) (irow(i) ,i= 1 , neq)
           read(8) (icoln(i), i = 1 , nc)
           read(8)( diag(i), i = 1 , neq )
           read(8)( offdia(i), i = 1, nc )
           read(8)( b(i), i = 1 , neq )
           icount = 0
        do 1 i = 1 , neq
1          diag(i) = diag(i) * x(i)
        do 2 i = 1 , neq - 1
           nonz = irow(i)
        do 2 j = 1 , nonz
           icount = icount + 1
           locate= icoln(icount)
           diag(i) = diag(i) + offdia(icount)*x(locate)
2          diag(locate)=diag(locate)+offdia(icount)*x(i)
           enorm = 0.0
           fnorm = 0.0
           snorm = 0.0
        do 3 i = 1 , neq
           diag(i) = diag(i) - b(i)
           enorm = enorm + diag(i) * diag(i)
           fnorm = fnorm + b(i)*b(i)
3          snorm = snorm + diag(i)*x(i)
           write(*,*)'* ABSOLUTE error norm = ',sqrt(enorm)
           relerr = sqrt(enorm/fnorm)
           write(*,*) '* RELATIVE to load = ',relerr
           write(*,*) '* STRAIN ENERGY error norm = ',snorm
           return
           end
       subroutine CSMIN(a,b,maxa,irowl,icolh,neq,nterms,
     +         irow,icoln,nc,maxbw,iin,locrow,iavebw)
        dimension a(*),b(*),maxa(*),irowl(*),icolh(*),irow(*),ic
```

```
c.....read binary file COEFS.COLM output by iter(sipr=-2)...
      open(unit=8,file='COEFS.COLM',form='unformatted',
     +     access='sequential',status='old')
      read(iin) neq,neq2,nc,nc2,jdof,jt,ndof
      read(iin) (irow(i), i = 1,neq)
      read(iin) (icoln(i), i=1,nc)
c.....initialize column heights.................................
      loop =9
      do 100 i = 1, neq
100     icolh(i) = 0.0
      icount = 1
      do 110 i = 1, neq-1
        do 110 j =1, irow(i)
          jcol = icoln(icount)
          nowht = jcol - i
         if (nowht.gt.icolh(jcol)) icolh(jcol)=nowht
110       icount = icount+1
c.....find the row-lengths...............................
      isegl = loop*neq/loop
      jcount = 0
      icount = 1
      do 120 i = 1, isegl, loop
        jcount = jcount + irow(i)
        if (icoln(jcount).gt.icount) icount=icoln(jcount)
        do 130 j = i+1, i+loop-1
          jcount =jcount + irow(j)
130       if (icoln(jcount).gt.icount) icount=icoln(jcount)
        do 140 j = i,i+loop-1
140       irowl(j) = icount - j
120     continue
      do 150 i = isegl+1,neq
150     irowl(i) = neq - i
c.....locate diagonal elements in vector {a}...............
      maxa(1) = 1
      do 160 i =1, neq
160     maxa(i+1) = maxa(i) + irowl(i) +1
      icount = 1
      do 170 i = 1, neq-1
        do 170 j = 1, irow(i)
          jcol = icoln(icount)
          locate = maxa(i) +jcol - i
          icoln(icount) = locate
170       icount =icount +1
      nterms = maxa(neq+1) - 1
      do 180 i = 1, nterms
180     a(i) = 0.0
      read(iin)  (a(maxa(i)), i=1,neq)
      read  (iin)  (a(icoln(i)),i=1,nc)
      read( iin)  (b(i), i=1,neq)
c.....find maximum and average bandwidths...........
      maxbw = 0
      iavebw = 0
      do 190 i = 1, isegl, loop
        if (irowl(i) .gt. maxbw) then
          maxbw = irowl(i)
          locrow = i
        endif
190     iavebw = iavebw + loop*irowl(i) - (loop)*(loop-1)/2
      do 200 i = isegl+1,neq
200     iavebw = iavebw + irowl(i)
      iavebw = iavebw/(neq+1)
      maxbw =maxbw + 1
      return
      end
      subroutine TOCSM(x,irowl,icoln,b,u,irtoj,iin,nat)
      dimension irowl(*),icoln(*),b(*),u(*),x(*),irtoj(*)
      character*40 libnam
      common /constr/jt,jdf,jddf,inex(6),mexin(6),ksym(3),q,qq
c       convert static displacements calculated by pvsolve
c       to csm testbed joint reference frame for [k]{u}={f}
c       assume each node has 6 degrees-of-freedom ( i.e.,
c       u(14) is the 2nd dof of node #3) and
c       jdof = number of joints * number of dof per joint
      read '(a)',libnam
      nu = lmopen('old',0,libnam,0,1000)
      call dal(nu,11,jt,18,-1,lseq,ierr,nwds,ne,lb,ityp,
     + 4hJDF1,4hBTAB,1,8)
c......read COEFS.COLM as in subroutine NORM.............
      rewind iin
      read(iin) n,n,nc,nc,jdof,jt,ndof
      if(nat.ge.2*jdof.and.nat.ge.ncoef) then
      read (iin) (irowl(i),i=1,n)
      read(iin) (icoln(i),i=1,nc)
      read(iin) (b(i),i=1,n)
      read(iin) (b(i),i=1,nc)
      read(iin) (b(i),i=1,jdof)
c......COEFS.COLM stores joint-to-row before row-to-joint.
c......only row-to-joint info. needed, so storage reused...
      read(iin) (jtorj(i),i=1,2*jdof)
      else
      write(*,*) 'error in TOCSM: insufficient memory'
      endif
c......initialize joint displacement...................
      do 4 i=1,jdof
4       u(i)=0.
      do 1 i=jdof+1,jdof+n
        locate = irtoj(i)
1       u(locate) = x(i-jdof)
c.....put prescribed displacements in vector {u}.....
      do 2 i = jdof+n+1,2*jdof
            if(irtoj(i).ne.0) then
            locate = irtoj(i)
            u(locate)= b(i-jdof)
           endif.
2     continue
c.....write displacements for first 3 joint locations
      njoint = jdof/6
      do 3 i=1,3
        i1 = (i-1)*6 + 1
        i2 = i*6
3      write(6,5) i,(u(j),j=i1,i2)
5      format('jt',i5,' disp=',6e11.3)
c.....put displacements in csm testbed library file
c      'libnam' (load set 1, constraint set 1)
        iset = 1
        ncon = 1
        nrhs = 1
        nwds = jdof*nrhs
      call gmsign('PVSOLVE')
      call dal(nu,0,0,0,1,lseq,ierr,nwds,jt,jdf,-1,
     + 4hSTAT,4hDISP,iset,ncon)
      call rio(nu,1,2,lseq,1,nrhs,u(1),nwds,-1,jt)
      call gmclos(nu,0,9999)
        return
        end
```

The command file to compute static displacements for the research aircraft and space shuttle SRB on the Cray Y-MP using 1 to 8 processors follows. The first statements specify the UNIX C-shell is used and the maximum number of processors (NCPUS) that may be requested is 8. The stiffness matrix data (COEFS.COLM) and program (pvsolve) are then copied to the solid state disk ($WRKDIR). Using the hardware performance monitor, hpm, to count operations, times and

MFLOPS, the displacements for the aircraft and SRB are then calculated by **pvsolve** on 8,4,2 and 1 processors. The results are appended to the file 'out' which, upon completion, is copied to the home directory:

```
#!/bin/sh
NCPUS=8
export NCPUS
cd $WRKDIR
date >out
cp /u/ra/storaasl/nasp/COEFS.COLM .
cp /u/ra/storaasl/srb/pvsolve .
date >>out
ja
hpm -g0 -d forcerun pvsolve 8  >>out 2>&1
hpm -g0 -d forcerun pvsolve 4  >>out 2>&1
hpm -g0 -d forcerun pvsolve 2  >>out 2>&1
hpm -g0 -d forcerun pvsolve 1  >>out 2>&1
date >>out
cp /scr5/storaasl/srb/COEFS.COLM .
date >>out
hpm -g0 -d forcerun pvsolvesrb 8  >>out 2>&1
hpm -g0 -d forcerun pvsolvesrb 4  >>out 2>&1
hpm -g0 -d forcerun pvsolvesrb 2  >>out 2>&1
hpm -g0 -d forcerun pvsolvesrb 1  >>out 2>&1
date >>out
cp out $HOME
```

Pvsolve is run in the CSM Testbed[18] structural analysis software to compute the static displacements for the SRB using the *spawn command in the following runstream using four Cray Y-MP processors:

```
testbed
*open 1 srb.l01
[xqt iter
reset sipr = -2
stop
*close 1
*spawn pvsolve srb.l01 4
*open 1 srb.l01 /old
[xqt vprt
PRINT STAT DISP
[xqt gsf
[xqt psf
[xqt exit
```

The 'iter' reset option bypasses the lengthy solution process and just formats the data for **pvsolve**. **Pvsolve** computes the static displacements and writes them to the data set STAT.DISP.1.1 in the CSM Testbed library srb.l01. The stresses are then calculated and printed based on the displacements calculated by **pvsolve**. The pvsolve code above is compiled using **force** producing the executable file, pvs. The pvsolve in the *spawn command is the following script that resides in the directory containing the CSM Testbed executable files:

```
forcerun pvs $2 <<EOF
$1
EOF
```

9. References

[1]Mackintosh, A. R., "The First Electronic Computer", *Physics Today*, March 1987, pp. 25-32.

[2]Ortega, J. M., *Introduction to Parallel and Vector Solution of Linear Systems*, Plenum Publishing Corporation, New Jersey, 1988.

[3]Utku, S., Salama, M., and Melosh, R., "Concurrent Factorization of Positive Definite Banded Hermitian Matrices", *International Journal of Numerical Methods in Engineering*, Vol. 23, 1986, pp. 2137-2152.

[4]Farhat, C., Wilson, E., and Powell, G., "Solution of Finite Element Systems on Concurrent Processing Computers", *Engineering Computing*, Vol. 2, 1987, pp. 157-165.

[5]Chen, S., Dongarra, J., and Hsiung, C., "Multiprocessing Linear Algebra Algorithms on the Cray XMP-2: Experiences With Small Granularity", *Journal of Parallel Distributed Computing*, Vol. 1, 1984, pp. 22-31.

[6]Dongarra, J. J., Gustafson, F. G., and Karp, A., "Implementing Linear Algebra Algorithms for Dense Matrices on a Vector Pipeline Machine", *SIAM Review*, Vol. 26, No. 1, January, 1984.

[7]Ashcraft, C. C., Grimes, R. G., Lewis, J. G., Peyton, B. W., and Simon, H. D., "Progress in Sparse Matrix Methods for Large Linear Systems on Vector Supercomputers", *The International Journal of Supercomputer Applications*, Vol. 1, No. 4, Winter 1987, pp. 10-30.

[8]Poole, E. L., and Overman, A. L., "The Solution of Linear Systems of Equations with a Structural Analysis Code on the NAS Cray 2", NASA CR 4159, Dec. 1988.

[9]Storaasli, O. O., and Bergan, P. G., "Nonlinear Substructuring Method for Concurrent Processing Computers", *AIAA Journal*, Vol. 25, No. 6, June 1987, pp. 871-876.

[10]Law, K., "A Parallel Finite Element Solution Method", *Computers and Structures*, Vol. 23, No. 6, 1986, pp. 845-858.

[11]Farhat, C. , and Wilson, E. L., "A Parallel Active Column Equation Solver", *Computers and Structures*, Vol. 28, 1988, pp. 289-304.

[12]Storaasli, O. O., Poole, E. L., Ortega, J. M., Cleary, A., and Vaughan, C., "Solution of Structural Analysis Problems on a Parallel Computer", *Proceedings of the AIAA/ASME/ASCE/AHS 29th Structures, Structural Dynamics and Materials Conference*, Williamsburg, VA, April 18-20, 1988, pp. 596-605, AIAA Paper No. 88-2287.

[13]Storaasli, O. O., Bostic, S. W., Patrick, M., Mahajan, U., and Ma, S., "Three Parallel Computation Methods for Structural Vibration Analysis", *Proceedings of the AIAA/ ASME/ASCE/AHS 29th Structures, Structural Dynamics and Materials Conference*, Williamsburg, VA, Apr. 18-20, 1988, pp. 1401-1411, AIAA Paper No. 88-2391.

[14]Nguyen, D. T., Shim, J. S., and Zhang, Y., "The Component-Mode Method in a Parallel Computer Environment", *Proceedings of the AIAA/ASME/ASCE/AHS 29th Structures, Structural Dynamics and Materials Conference*, Williamsburg, VA, April 18-20, 1988, pp. 1705-1710, AIAA Paper No. 88-2438.

[15]Nguyen, D. T., and Niu, K. T., "A Parallel Algorithm for Structural Sensitivity Analysis on the FLEX/32 Multicomputer", *Proceedings of the 6th ASCE Structures Congress*, Orlando, FL, August 17-20, 1987, pp. 98-112.

[16]Storaasli, O. O., Nguyen, D. T., and Agarwal, T. K., "The Parallel Solution of Large-Scale Structural Analysis Problems on Supercomputers", *Proceedings of the AIAA/ASME/ASCE/AHS 30th Structures, Structural Dynamics and Materials Conference*, Mobile AL, April 3-5, 1989, pp. 859-867. Paper No. 89-1259 (to appear in *AIAA Journal*, Sept. 1990)

[17]Jordan, H. F., Benten, M. S., Arenstorf, N. S., and Ramann, A. V., "Force User's Manual: A Portable Parallel FORTRAN", NASA CR 4265, January, 1990.

[18]Stewart, C. B.(compiler), "The Computational Structural Mechanics Testbed User's Manual", NASA TM-100644, October 1989.

[19]George, A. and W-H Liu, J., *Computer Solution of Large Sparse Positive Definite Systems*. Prentice Hall, Inc., Englewood Cliffs, NJ, 1981.

[20]Bathe, K. J., *Finite Element Procedures in Engineering Analysis*, Prentice Hall, Inc., New York, 1982.

[21]Robins, W. A. et al., "Concept Development of a Mach 3.0 High-Speed Civil Transport", NASA TM 4058, Sept. 1988.

[22]Knight, N. F., McCleary, S. L., Macy, S. C., and Aminpour, M. A., "Large Scale Structural Analysis: The Structural Analyst, The CSM Testbed, and The NAS System", NASA TM-100643, March 1989.

[23]Knight, N. F., Gillian, R. E., and Nemeth, M. P., "Preliminary 2-D Shell Analysis of the Space Shuttle Solid Rocket Boosters", NASA TM-100515, 1987.

[24]Simon, H., Vu, P. and Yang, C., "Performance of a Supernodal General Sparse Solver on the Cray Y-MP: 1.68 GFLOPS with Autotasking", Scientific and Computing Analysis Division Report SCA-TR-117, Boeing Computer Services, Seattle, WA, March, 1989.

[25]Storaasli, O., Nguyen, D., and Agarwal, T., "Force on the Cray Y-MP", */u/nas/news The Numerical Aerodynamic Simulation Program Newsletter*, NASA Ames Research Center, Vol. 4, No. 7, July 1989, pp. 1-4.

[26]Storaasli, O. O., "New Equation Solver for Supercomputers", */u/nas/news The Numerical Aerodynamic Simulation Program Newsletter*, NASA Ames Research Center, Vol. 5, No. 1, January 1990, pp. 1-3.

NASA

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-102614		

4. Title and Subtitle	5. Report Date
A Parallel-Vector Algorithm for Rapid Structural Analysis on High-Performance Computers	April 1990
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Olaf O. Storaasli Duc T. Nguyen Tarun K. Agarwal	
	10. Work Unit No. 505-63-01-10
9. Performing Organization Name and Address NASA Langley Research Center Hampton, VA 23665-5225	11. Contract or Grant No.
	13. Type of Report and Period Covered Technical Memorandum
12. Sponsoring Agency Name and Address National Aeronautics and Space Administration Washington, DC 20546-0001	14. Sponsoring Agency Code

15. Supplementary Notes

Expanded version of AIAA Paper No. 90-1149 presented at the AIAA/ASME/ASCE/AHS 31st Structures, Structural Dynamics and Materials Conference, Long Beach, CA April 2-4, 1990. Part of this work was supported by NASA Grant NAG-1-858 with Old Dominion University (ODU).

16. Abstract

A fast, accurate Choleski method for the solution of symmetric systems of linear equations is presented. This direct method is based on a variable-band storage scheme and takes advantage of column heights to reduce the number of operations in the Choleski factorization. The method employs parallel computation in the outermost DO-loop and vector computation via the "loop unrolling" technique in the innermost DO-loop. The method avoids computations with zeros outside the column heights, and as an option, zeros inside the band. The results for two large-scale structural analyses performed on supercomputers, demonstrates the accuracy and speed of the method. The listing of the computer program, PVSOLVE, and a simple example with input data are contained in Appendices B and C. The use of PVSOLVE for parallel equation solution in a stand-alone mode as well as its use in the CSM Testbed structural analysis system is described in Appendix C.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
structural analysis linear equations simultaneous equations	FEDD Subject Category 39

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	19	

NASA FORM 1626 OCT 86

APPENDIX B: Parallel FORTRAN Listing of Subroutine Golden Block

```
      Force GOLDB of NP ident ME
      Shared REAL ALPHA(30),FVALUE(30)
      Shared REAL A,EPS,AA,FMIN,DELTA
      Shared INTEGER K,L,IMAX
      Shared REAL T1(10),T2(10),TT(10)
      Shared REAL TMAX
      End declarations
      Barrier
C      K=4*NP
      READ(5,*) A,DELTA,K,EPS
      L=30
      write(6,*) a,delta,k,l,eps
      End barrier
      T1(ME)=Tsecnd( )
      Forcecall GOLD(K,A,DELTA,FMIN,AA,ALPHA,FVALUE,EPS,L)
      T2(ME)=Tsecnd( )
      TT(ME)=T2(ME)-T1(ME)
      Barrier
C      WRITE(6,*) 'MIN. F=',FMIN
C      WRITE(6,*) 'ALPHA =',AA,'with EPS=',EPS
      IMAX=Ismax(NP,TT,1)
      TMAX=TT(IMAX)
      WRITE(6,*) 'Time used=',TMAX
      End barrier
      Join
      END
C **************************************************
      Forcesub GOLD(K,A,DELTA,FMIN,AA,ALPHA,FVALUE,EPS,L) of NP ident ME
      REAL ALPHA(L),FVALUE(L)
      REAL A,EPS,AA,FMIN
      Private INTEGER J
      INTEGER K
      Shared INTEGER ICOUNT,KK,KK1,KK2,II,IMIN,IQM1,IQP1,I
      Shared REAL CC,GR,DB,CCC,BK,SGR,B,A0
      End declarations
      Barrier
      CC=FLOAT(K**2+4*K)
      SGR=0.5*SQRT(5.0)+0.5
      GR=0.5*(FLOAT(K)+SQRT(CC))
      KK = 2*K
      KK1=kk+1
      kk2=kk+2
       ALPHA(1)=DELTA
      DO 30 I=2,12
       ALPHA(I)=ALPHA(I-1)+DELTA*(SGR**(I-1))
       write(6,*) 'alpha(i)',i,alpha(i)
 30   CONTINUE
      End barrier
      Presched DO 40 J=1,12
       CALL FUNCT(ALPHA(J),FVALUE(J))
      write(6,*) 'alpha,fvalue',alpha(j),fvalue(j)
 40   End Presched DO
      Barrier
      IMIN=ISMIN(12,FVALUE,1)
      IQM1=IMIN-1
      IQP1=IMIN+1
      A=ALPHA(IQM1)
      B=ALPHA(IQP1)
      A0=A
      write(6,*) 'a,b',a,b
```

```
      DB=B-A
      BK=DB/K
      FVALUE(1)=FVALUE(IQM1)
      FVALUE(KK1)=FVALUE(IQP1)
      FVALUE(KK2)=100000.00
      ICOUNT=1
      End barrier
 10   CONTINUE
      Barrier
      ALPHA(1)=A
      ALPHA(2)=A+(1.0/GR)**ICOUNT*DB
      II=ABS(1-ICOUNT)
      CCC=BK/(GR**II)
      DO 20 I=3,KK1,2
      ALPHA(I)=ALPHA(I-2)+CCC
      ALPHA(I+1)=ALPHA(I-1)+CCC
 20   CONTINUE
      End barrier
      Presched DO 25 J=2,KK1
      CALL FUNCT(ALPHA(J),FVALUE(J))
 25   End presched DO
      Barrier
      IMIN=ISMIN(KK2,FVALUE,1)
      FMIN=FVALUE(IMIN)
      AA=ALPHA(IMIN)
      WRITE(6,*) 'alpha=',AA,'FMIN=',FVALUE(IMIN)
        IQP1=IMIN+1
        IQM1=IMIN-1
      End barrier
      IF(ABS(A0-ALPHA(IMIN)).LT. EPS)  GO TO 100
      Barrier
      A=ALPHA(IQM1)
      B=ALPHA(IQP1)
      A0=ALPHA(IMIN)
      FVALUE(KK2)=FVALUE(IQP1)
      FVALUE(1)=FVALUE(IQM1)
      ICOUNT=ICOUNT+1
      End barrier
      GO TO 10
 100  RETURN
      END
C *****************************************************
      SUBROUTINE FUNCT(T,F)
      REAL T,F
      REAL sign,fact,value
      INTEGER I,j
C      do 40 nn=1,100
C      F=2.0-4.0*T+EXP(T)
C       F=COS(T)
C 40     f=f+f
c      f=2.0-4.0*t+exp(t)
      F=1.0
      sign=1.0
      do 10 i=2,600,2
      sign=sign*(-1.0)
      fact=1.0
      value=1.0
      do 20 j=1,i
        fact=fact*j
 20     value=value*t
```

```
      f=f+sign*value/fact
10    continue
      RETURN
      END
```

APPENDIX C: Parallel FORTRAN Listing of Subroutine BFGS

```
C THIS PROGRAM IS WRITTEN ON JULY 20 1989        BY : MAJDI BADDOURAH
C THIS PROGRAM WILL SOLVE UNCONSTRAINED NONLINEAR OPTIMIZATION
C USING B F G S   M E T H O D

C ------------------- B F G S   M E T H O D ----------------------

      Force MAB of NP ident ME
      Shared DOUBLE PRECISION H(1000000),C(800),D(800),X(800),CS(800)
      Shared DOUBLE PRECISION H1(100000)
      Shared DOUBLE PRECISION F(800),HH(800),G(800),S(800),Y(800)
      Private DOUBLE PRECISION CP(800)
      Shared INTEGER MAXA(800),ICOLH(800),ISWTCH
      Shared INTEGER IFLAG,IW,IR,NTERMS,N,MXNITB,NBW,MXNITS,JFLAG
      Shared DOUBLE PRECISION TOLBFG,TOLSOR,THETIM,TIMAX,PI,DIV
      REAL*8 TIME1(16),TIME2(16),TIMER
      Shared DOUBLE PRECISION TIMEE1(16),TIMEE2(16)
      Shared LOGICAL TYPE1,TYPE2
      End declarations
      Barrier
      DIV = 1000000.
      PI = ACOS(-1.0)
      IR = 5
      IW = 6
      WRITE(6,*)'  ENTER NUMBER OF EQUATIONS  & 1 FOR ALFA 2 FOR NO ALFA'
      WRITE(6,*)'  ENTER ISWITCH '
      READ(5,*) N,JFLAG,ISWTCH
      MXNITB = 500
      MXNITS = 300
      NBW = N
      TOLBFG = 1.0E-01
      TOLSOR = 1.0E-05
      WRITE(IW,*)' ENTER TOL FOR BFGS   TOL FOR SOR '
      READ(5,*) TOLBFG,TOLSOR
      NTERMS = 0
      ISUM = 1
      MAXA(1) = 1
      DO 10 I = 1 , N , 2
      ICOLH(I) = 0
      ICOLH(I+1) = 1
      NTERMS = NTERMS + ICOLH(I)
C     ISUM = ISUM + ICOLH(I-1)
C     MAXA(I) = ISUM
C     ISUM = ISUM + ICOLH(I)
C     MAXA(I+1) = ISUM
10    CONTINUE
C     MAXA(N+1) = MAXA(N) + ICOLH(N) + 1
      NTERMS = NTERMS + N
      DO 11 I = 1, N
11    ICOLH(I) = I - 1
      CALL      ADD1(N,ICOLH,MAXA,NTERMS)
      WRITE(6,*)' NUMBER OF EQUATIONS = ',N
      WRITE(6,*)' NUMBER OF TERMS = ',NTERMS
C     WRITE(6,*)' COL HIEGHT = ',(ICOLH(I),I=1,N)
C     WRITE(6,*)' MAXA = ',(MAXA(I),I=1,N+1)
      End barrier
C     TIME1(ME) = SECOND()
C     TIME1(ME) = TSECND()
C     Critical TYPE1
      TIME1(ME) = TIMER()
      TIMEE1(ME) = TIME1(ME)
C     End critical
      Forcecall BFGSOP (IW,IR,N,NTERMS,H,H1,C,D,X,CP,CS,Y,S,MAXA,ICOLH,
     & MXNITB,MXNITS,TOLBFG,TOLSOR,NBW,F,HH,G,JFLAG,ISWTCH,DIV,PI)
C     TIME2(ME) = SECOND()
      TIME2(ME) = TIMER()
      TIMEE2(ME) = TIME2(ME)
```

```
      Barrier
      TIMAX = 0.0
C     DO 120 I = 1 , N
C120   WRITE(IW,*)'  X( ',I,' ) = ',X(I)
      WRITE(IW,*) 'X(1) = ',X(1)
      WRITE(IW,*) 'X( ',N,' ) = ',X(N)
      DO 130 I = 1 , NP
      THETIM = (TIMEE2(I) - TIMEE1(I)) / 1000000.
      WRITE(IW,*)' PROCESS NO : ',I,'   TIME = ',THETIM
      TIMAX = MAX (TIMAX,THETIM)
130   CONTINUE
      WRITE(IW,*) ' THE MAX TIME = ',TIMAX
C     WRITE(6,*)' NP = ',NP , '    TIME = ',TIME2(I) - TIME
      End barrier
      Join
      END

      Forcesub BFGSOP (IW,IR,N,NTERMS,H,H1,C,D,X,CP,CS,Y,S,MAXA,ICOLH,
     & MXNITB,MXNITS,TOLBFG,TOLSOR,NBW,F,HH,G,JFLAG,ISWTCH,DIV,PI)
     & of NP ident ME
      DOUBLE PRECISION H(NTERMS),C(N),D(N),X(N),CP(N),CS(N),Y(N)
      DOUBLE PRECISION S(N),F(N),HH(N),G(N),H1(NTERMS)
      DOUBLE PRECISION TOLBFG,TOLSOR,DIV,PI
      INTEGER MAXA(N+1),ICOLH(N)
      Shared DOUBLE PRECISION W,ALFA,SUMS1,SUMS2,SUMS3,DELTA,CONST,CONST1
      REAL*8 TC1(16),TC2(16),TS1(16),TS2(16),TALFP(16),TALFW(16)
      Shared DOUBLE PRECISION  TCE1(16),TCE2(16),TSE1(16),TSE2(16)
      Shared DOUBLE PRECISION  TALF1(16),TALF2(16)
      Private DOUBLE PRECISION SUMP1,SUMP2
      Private INTEGER ITEMP
      Shared LOGICAL TYPE1,TYPE2,TYPE3
      End declarations
      SUMP1 = 0.0
      SUMP2 = 0.0
      DIV = 1000000.0
      TSE1(ME) = 0.0
      TSE2(ME) = 0.0
      TCE1(ME) = 0.0
      TCE2(ME) = 0.0
      TALF1(ME) = 0.0
      TALF2(ME) = 0.0

      Barrier
C --->  READ Initial guess for BFGS
C     Write(6,*) ' READ Initial guess for BFGS , Two values '
C     READ(5,*)CONST , CONST1
      DELTA = .01
      SUMS1 = 0.0
      SUMS2 = 0.0
      ALFA = 1.00
      W = 1.0
      DO 10 I = 1,NTERMS
      H(I) = 0.0
10    CONTINUE
      DO 20 I = 1,N
      H(MAXA(I)) = 1.0
20    CONTINUE
      End barrier

      Barrier
      End barrier
C ---> Initial guess for BFGS

      Presched do 11 I = 1 , N,2
      X(I) = .10
      X(I+1) =  .40
```

```
11      End Presched do

        Barrier
        End barrier

        Forcecall FSTD (N,C,X,NBW)
C       write(6,*)'c1(i) c2(i)',c(1),c(2)

        Barrier
        End barrier

        Presched do 8 I = 1 , N
        D(I) = - C(I)
        C(I) = -C(I)
8       End Presched do
C       write(6,*)'d1(i) d2(i)',d(1),d(2)

        Barrier
        End barrier

C ------------------  ITTERATION START AT THIS LEVEL  -------------------

        DO 100 ICONT = 1 , MXNITB


        Barrier
        DO 30 I = 1,N
        SUMS3 = SUMS3 + C(I) * C(I)
30      CONTINUE
        SUMS3 = DSQRT(SUMS3)
        write(iw,*)' T H E   N O R M   = ',SUMS3
        SUMS1 = 0.0
        SUMS2 = 0.0
        End barrier

        Barrier
        End barrier

        IF( SUMS3 .LT. TOLBFG ) GO TO 110
        TALFP(ME) = TIMER()
        TALF1(ME) = TALF1(ME) + TALFP(ME)
        Barrier
        IF(JFLAG.EQ. 1) THEN
C       CALL       ALFAQ (N,X,D,G,ALFA,TOLBFG,DELTA,C)
        CALL       GOLDEN (N,X,D,G,ALFA,.000001,DELTA,C)
C       WRITE(6,*)' A L F A  ----------> ',ALFA
        ENDIF
        End barrier

        TALFW(ME) = TIMER()
        TALF2(ME) = TALF2(ME) + TALFW(ME)
        Presched do 60 I = 1,N
        X(I) = X(I) + ALFA *  D(I)
        Y(I) = C(I)
        SUMP2 = SUMP2 - C(I) * D(I)
60      End presched do

C       write(6,*)'x1(i) x2(i)',x(1),x(2)
        Barrier
        End barrier

        Critical TYPE1
        SUMS2 = SUMS2 + SUMP2
        SUMP2 = 0.0
        End critical
```

```
      Barrier
      End barrier

      Presched do 91 I = 1 , N
      ITEMP = I
      DO 81 J = MAXA(I) , MAXA(I) + ICOLH(I)
      H(J) = H(J) + C(I) * C(ITEMP) / SUMS2
81    ITEMP = ITEMP - 1
91    End presched do

      Barrier
      End barrier

      Forcecall FSTD (N,C,X,NBW)

      Barrier
      End barrier

      Presched do 70 I = 1,N
      Y(I) = C(I) + Y(I)
      SUMP1 = SUMP1 + Y(I) * ALFA * D(I)
70    End presched do

      Barrier
      End barrier

      Critical TYPE1
      SUMS1 = SUMS1 + SUMP1
      SUMP1 = 0.0
      SUMS3 = 0.0
      End critical

      Barrier
      End barrier

      Presched do 90 I = 1 , N
      ITEMP = I
      DO 80 J = MAXA(I) , MAXA(I) + ICOLH(I)
      H(J) = H(J) + Y(I) * Y(ITEMP) / SUMS1
80    ITEMP = ITEMP - 1
90    End presched do

      Forcecall FSTD (N,C,X,NBW)

      Barrier
      End barrier

      Presched do 92 I = 1 , N
      C(I) = -C(I)
92    End Presched do

      Barrier
      End Barrier
      IF( ICONT .LT. ISWTCH ) THEN
      Presched do  31 i = 1 , nterms
      H1(i) = H(i)
31    End presched do

      Presched do 32 I = 1 , N
      D(I) = C(I)
32    End presched do
      ENDIF

      Barrier
      End barrier
```

```
C       write(6,*)'C(I) = ',(C(I),I=1,N)
C       write(6,*)'D(I) = ',(D(I),I=1,N)
C       write(6,*)' h(I) = ',(h(I),i=1,nterms)

        IF( ICONT .LT. ISWTCH ) THEN
        TC1(ME) =  TIMER()
        TCE1(ME) = TCE1(ME) + TC1(ME)
        Forcecall FF(H1,MAXA,D,N,1,ICOLH)
        Forcecall FF(H1,MAXA,D,N,2,ICOLH)
        TC2(ME) =  TIMER()
        TCE2(ME) = TCE2(ME) + TC2(ME)
        ELSE
        TS1(ME) =  TIMER()
        TSE1(ME) = TSE1(ME) + TS1(ME)
        Forcecall SOR1(N,NTERMS,H,C,D,CP,CS,MAXA,NBW,TOLSOR,MXNITS,W,ICOLH)
        TS2(ME) =  TIMER()
        TSE2(ME) = TSE2(ME) + TS2(ME)
        ENDIF


C       write(6,*)'D(I) = ',(D(I),I=1,N)
        Barrier
C        DO 140 I = 1 , N
C140      WRITE(IW,*) '  X( ',I,' ) = ',X(I)
        End barrier

100     CONTINUE

110     CONTINUE

        Barrier
        WRITE(IW,*)' NUMBER OF ITTERATIONS = ',ICONT
C       DO 120 I = 1 , N
C120     WRITE(IW,*) '  X( ',I,' ) = ',X(I)
        DO 130 I = 1 , NP
         TIMEC = (TCE2(I) - TCE1(I)) / DIV
         TIMES = (TSE2(I) - TSE1(I)) / DIV
         TIMEA = (TALF2(I) - TALF1(I)) / DIV
         WRITE(6,*)' CHOL TIME @ PROC # ',I,' TIME = ',TIMEC
         WRITE(6,*)' SOR  TIME @ PROC # ',I,' TIME = ',TIMES
         WRITE(6,*)' ALFA TIME @ PROC # ',I,' TIME = ',TIMEA
130        CONTINUE
C       Write(6,*)' H(I) =',(H(I),I= 1,NTERMS)
        End barrier
        RETURN
        END

        Forcesub FSTD (N,F,X,NBW) of NP ident ME
        DOUBLE PRECISION F(N),X(N)
        Private INTEGER MSTART,MEND
        Shared INTEGER NBWT
        Private DOUBLE PRECISION SUM
        End declarations
        NBWT = NBW - 1
        Presched do 20 I = 1 , N
        F(I) = 0.0
        SUM = 0.0
        MEND = MIN(N,NBWT+I)
        IF ( I .LT. NBW ) THEN
        MSTART = 1
        ELSE
        MSTART = I - NBWT
        ENDIF
        DO 10 J = MSTART , MEND
        IF( I .EQ. J ) THEN
        F(I) = F(I) + 2.0 * X(I) * X(J) * X(J)
```

```
      SUM = SUM + 2.0
      ELSE
      F(I) = F(I) + (1.0/(I+J)) * X(I) * X(J) * X(J)
      SUM = SUM + (1.0/(I+J))
      ENDIF
10    CONTINUE
      F(I) = F(I) - SUM
20    End presched do
      RETURN
      END



      SUBROUTINE NEWF (N,F,X,NBW)
      DOUBLE PRECISION F(N),X(N)
      INTEGER MSTART,MEND
      INTEGER NBWT
      DOUBLE PRECISION SUM
      NBWT = NBW - 1
      do 20 I = 1 , N
      F(I) = 0.0
      SUM = 0.0
      MEND = MIN(N,NBWT+I)
      IF ( I .LT. NBW ) THEN
      MSTART = 1
      ELSE
      MSTART = I - NBWT
      ENDIF
      DO 10 J = MSTART , MEND
      IF( I .EQ. J ) THEN
      F(I) = F(I) + 2.0 * X(I) * X(J) * X(J)
      SUM = SUM + 2.0
      ELSE
      F(I) = F(I) + (1.0/(I+J)) * X(I) * X(J) * X(J)
      SUM = SUM + (1.0/(I+J))
      ENDIF
10    CONTINUE
      F(I) = F(I) - SUM
20    End presched do
      RETURN
      END

      Forcesub FSTDD11     (N,C,X) of NP ident ME
      DOUBLE PRECISION C(N),X(N)
      DOUBLE PRECISION PI
      Shared  DOUBLE PRECISION PII
      Private  INTEGER TEMP10
      End declarations
      PII = ACOS( -1.0 )
      Presched do 20 I = 1 , N
      C(I) = 1.0
      DO 10 J = 1 , N
      IF(I .EQ. J) THEN
      C(I) = C(I) * DCOS(X(I))
      ELSE
      C(I) = C(I) * DSIN(X(J))
      ENDIF
      TEMP10 = FLOAT(I)
      C(I) = C(I) +  X(I) -  TEMP10 * PII
10    CONTINUE
20    End presched do
      RETURN
      END

      Forcesub FSTD23    (N,C,X) of NP ident ME
      DOUBLE PRECISION C(N),X(N)
```

```
      End declarations
      Presched do 10 I = 1 , N , 2
      C(I) = 10.0 * X(I) + 2.0 * X(I+1)
      C(I+1) = 2.0 * X(I) + 2.0 * X(I+1)
10    End presched do
      RETURN
      END

      SUBROUTINE  FUNCT   (N,X,SUM,C)
      DOUBLE PRECISION X(N),SUM,C(N)
      SUM = 0.0
      DO 10 I = 1 , N
      SUM = SUM + (X(I)**4) / 2.0
10    CONTINUE
      DO 20 I = 1 , N - 1
      DO 20 J = I+1 , N
      SUM = SUM + (   (X(I)**2) * (X(J)**2) /(2.0*(I+J))    )
20    CONTINUE

      DO 40 I = 1 , N
      SUMM2 = 0.0
      DO 30 J = 1 , N
      IF( I .EQ. J ) THEN
      SUMM2 = SUMM2 + 2
      ELSE
      SUMM2 = SUMM2 + (1.0/(I+J))
      ENDIF
30    CONTINUE
      SUM = SUM - SUMM2 * X(I)
40    CONTINUE
C     DO 10 I  = 1 , N , 2
C     SUM = (X(I)**4)/2 +(X(I)**2) * (X(I+1)**2)/6.0 +(X(I+1)**4)/2.0
C    &  - 7.0*X(I)/3.0 -7.0*X(I+1)/3.0 + SUM
C10    CONTINUE
      RETURN
      END


      SUBROUTINE FUNCT9   (N,X,SUM)
      DOUBLE PRECISION X(N),SUM
      DO 10 I = 1 , N , 2
      SUM = 5.0 * (X(I)  **2) + 2.0 * X(I)  * X(I+1)  + X(I+1)**2 + 7
     & + SUM
10    CONTINUE
      RETURN
      END

      Forcesub FSTD19  (N,C,X) of NP ident ME
      DOUBLE PRECISION C(N),X(N)
      End declarations
      DO 10 I = 1 , N , 2
      C(I) = X(I+1) + 2.0 * X(I) - (X(I+1)**2) + EXP(X(I))
10    C(I+1) = X(I) - 2.0 * X(I) * X(I+1)
      RETURN
      END

C     SUBROUTINE FUNCT2  (N,X,SUM)
C     DOUBLE PRECISION X(N),SUM
C     SUM = 0.0
C     DO 10 I = 1 , N , 2
C     SUM = X(I) * X(I+1) + (X(I)**2) - X(I) * (X(I+1)**2) + EXP (X(I))
C    & + SUM
C 10    CONTINUE
C     RETURN
C     END
```

```
C        Forcesub FSTD1 (N,C,X) of NP ident ME
C        DOUBLE PRECISION C(N),X(N)
C        End declarations
c        C(1) = 400 *((X(1)**2) - X(2)) * X(1) - 2.0 * (1.0 - X(1))
C        C(2) = -200 * ((X(1)**2) -X(2))
C        RETURN
C        END

C       SUBROUTINE FUNCT1 (N,X,SUM)
C       DOUBLE PRECISION X(N),SUM
C       SUM = 100 * ( ( (X(1)**2)-X(2) ) **2) + ( (1.0 - X(1)) **2 )
C       RETURN
C       END

        SUBROUTINE GOLDEN (NR,B,S,D,ALFA,TOL,DELTA,C)
        DOUBLE PRECISION B(NR),C(NR),S(NR),D(NR)
        DOUBLE PRECISION ALFAA,ALFA,ALFAL,ALFAB,ALFAU,F1,F2,FA,FB,DELTA
        DELTA = .01
C        write(6,*) ' subroutine golden is used  after '
C        write(6,*)'delta tol, ',delta,tol
        TOL1=TOL
        ALFA=0.0
        F1=0.0
        DO 30 I=1,30
        ALFAA=ALFA
10      ALFA=ALFA+DELTA*(1.618**I)
        DO 20 J=1,NR
20      D(J)=B(J)+ALFA*S(J)
        F2=F1
        CALL       FUNCT (NR,D,F1,C)
C       write(6,*)'  f1,d1,d2',F1,D(1),D(2)
        IF(I.EQ.1) GO TO 30
        IF(F1.GT.F2) GO TO 40
30      CONTINUE
40      ALFAU=ALFA
        ALFAL=(ALFAA-.382*ALFAU)/.618
        ALFAB=.618*(ALFAU-ALFAL)+ALFAL
        DO 50 N=1,NR
50      D(N)=B(N)+ALFAB*S(N)
        CALL       FUNCT (NR,D,FB,C)
C       write(6,*)'  f2,d1,d2',Fb,D(1),D(2)
        DO 60 N=1,NR
60      D(N)=B(N)+ALFAA*S(N)
C       write(6,*)'  fa,d1,d2',Fb,D(1),D(2)
        CALL       FUNCT (NR,D,FA,C)
C*      WRITE(6,*)'ALFAL,ALFAU',ALFAL,ALFAU
        DO 90 KJ=1,100
C       WRITE(6,*)
C       WRITE(6,*)KJ
C       WRITE(6,*)
        IF(FA.LT.FB) THEN
        ALFAU=ALFAB
        ALFAB=ALFAA
        ALFAA=ALFAL+.382*(ALFAU-ALFAL)
        FB=FA
        DO 70 N=1,NR
70      D(N)=B(N)+ALFAA*S(N)
        CALL       FUNCT (NR,D,FA,C)
        ELSE IF(FA.GT.FB) THEN
        ALFAL=ALFAA
        FA=FB
        ALFAA=ALFAB
        ALFAB=ALFAL+.618*(ALFAU-ALFAL)
        DO 80 N=1,NR
80      D(N)=B(N)+ALFAB*S(N)
        CALL       FUNCT (NR,D,FB,C)
```

```
      ELSE IF(FA.EQ.FB) THEN
      ALFAL=ALFAA
      ALFAU=ALFAB
      ALFAA=ALFAL+.382*(ALFAU-ALFAL)
      ALFAB=ALFAL+.618*(ALFAU-ALFAL)
      ENDIF
      IF(DABS(ALFAA-ALFAB).LT.TOL1) GO TO 100
90    CONTINUE
100   ALFA=(ALFAA+ALFAB)/2
C     WRITE(6,*)'ALFA   ************',ALFA
      RETURN
      END

      SUBROUTINE ALFAQ (NR,B,S,D,ALFA,TOL,DELTA,C)
      DOUBLE PRECISION B(NR),S(NR),D(NR),C(NR)
      DOUBLE PRECISION ALFA,TOL ,DELTA,F1,F2,F3,CC1,CC2,CHEK,ALFA2,ALFA1
      DOUBLE PRECISION ALFA3
      INTEGER JCONT
      WRITE(6,*)'*********   SUBROUTINE ALFAQ IS USED   **********'
      WRITE(6,*) ' ALFA = ',ALFA,'   TOL = ',TOL,'  DELTA = ',DELTA
      JCONT=1
      ALFA1=0.0
      ALFA2=DELTA
10    ALFA3=2*ALFA2
      CALL        FUNCT (NR,B,F1,C)
      write(6,*) ' F1 = ',F1
      DO 20 I=1,NR
20    D(I)=B(I)+ALFA2*S(I)
      CALL        FUNCT (NR,D,F2,C)
      write(6,*) ' F2 = ',F2
      DO 30 I=1,NR
30    D(I)=B(I)+ALFA3*S(I)
      CALL        FUNCT (NR,D,F3,C)
      write(6,*) ' F3 = ',F3
      CHEK=((F3+F1)/2)-F2
      WRITE(6,35)F3,F2,F1,ALFA2,CHEK
35    FORMAT(7F10.3)
      IF(CHEK.LT.0.0) GO TO 40
      CC1=(4.0*F2-3.0*F1-F3)/(2*ALFA2)
      CC2=(F3+F1-2.0*F2)/(2.0*(ALFA2**2))
      IF(CC2.EQ.0.0) GO TO 50
      ALFA=-CC1/(2.0*CC2)
      GO TO 50
40    ALFA2=ALFA2+ALFA2*(1.618**JCONT)
      WRITE(6,*)' CHEK',CHEK
      IF(ABS(CHEK).LT.1.0D-40) THEN
      WRITE(6,*)' THE FUNCTION DOES NOT HAVE ANY MIN POINT'
      GO TO 60
      ENDIF
      JCONT=JCONT+1
      GO TO 10
60    STOP
50    RETURN
      END


      Forcesub FF(A,MAXA,B,NEQ,M,ICOLH) of NP ident ME
       DOUBLE PRECISION  A(1),B(1)
       INTEGER MAXA(1),ICOLH(1)
       Shared INTEGER jops(16)
       Private INTEGER I,J,K,L ,ipdig ,iloc,idig,ii,jj,i4,l1,i5,i6
       Private INTEGER IP1,IP2,IIp1,IIp2,IPloc,IPLOCa,IP3,IP4,IIP3,IIP4
       Private INTEGER Jp1,Jp2,Jjp1,JJp2
       Private DOUBLE PRECISION SUM1,SUM2,SUM3,SUM4,Y1(10000),Y2(10000)
       Private DOUBLE PRECISION SUM,TEMP
       Shared INTEGER IS1,IS2,N
```

```
      INTEGER NEQ,M,iops
      Shared Logical ialoc
c     Async DOUBLE PRECISION X(10001)
      Async DOUBLE PRECISION X(10001)
      End declarations
c..............................
c     Barrier
c     WRITE(6,*) 'MAXA(I)= ', (MAXA(I),I=1,NEQ+1)
c     WRITE(6,*) 'ICOLH(I)= ', (ICOLH(I),I=1,NEQ)
c     WRITE(6,*) 'A(NTERMS)= ', (A(I),I=1,MAXA(NEQ+1)-1)
c     WRITE(6,*) 'B(NEQ)= ', (B(I),I=1,NEQ)
c     End barrier
c..............................
      IF(M.EQ.1) THEN
C***********************************
      Presched DO 10 I=1,NEQ
      Void X(I)
 10   End Presched DO
      jops=0
      Barrier
       jops=0
      jops=jops+1
      Produce X(1)=A(1)
      is1=neq - 2*(neq/2)
      if (is1.eq.0) then
      is1=2
      if (maxa(3) .eq. 4) then
      a(3)=a(3)/a(1)
      a(2)=(a(2)-a(3)*a(3)*a(1))
      jops=jops+4
       Produce x(2)=a(2)
      else
       jops=jops+1
      Produce x(2)=a(2)
      endif
      endif
      End barrier
      Presched DO 20 I=is1+1,neq,2
        IP1=MAXA(I)
        IP2=MAXA(I+1)
        IIp1=IP1+I
        IIp2=IP2+i+1
        IPLOC=I-ICOLH(I)
        IIP3= ICOLH(I)-2*(ICOLH(I)/2)
        IPLOCA=IPLOC
        IF (IIP3.EQ.1) THEN
         IPLOCA = IPLOC +1
         ENDIF
        IIP4 = IPLOCA + 2* ((( ICOLH(I)/2) +1) /2) -1
        Copy X(IIP4) into TEMP
        IF (IIP3.EQ.1) THEN
         y1(iploc)=a(iip1-iploc)
         A(IIP1-IPLOC) = y1(IPLOC)/A(MAXA(IPLOC))
         y2(iploc) =a(iip2-iploc)
         A(IIP2-IPLOC) = y2(IPLOC) /A(MAXA(IPLOC))
         jops=jops+4
        ENDIF
25    continue
       DO 30 J=IpLOCa,IIP4,2
          Jp1=MAXA(J)
          JP2=MAXA(j+1)
          JJP1= JP1+J
          JJP2= JP2+J+1
          SUM1=0.0
          sum2=0.0
          sum3=0.0
```

```
          sum4=0.0
          ipdig=j - icolh(j)
          if (IPLOC .gt. IPDIG) IPDIG=IPLOC

c      if(A(Ip1)-SUM.LE.0.0) write(*,*) 'Matrix not pos. definite'
CDIR$ IVDEP
         DO 40 k=IpDIG,J-1
            sum1=sum1+a(jjp1-k)*Y1(k)
            sum2=sum2+a(jjp1-k)*y2(k)
            sum3=sum3+a(jjp2-k)*y1(k)
            sum4=sum4+a(jjp2-k)*y2(k)
 40      CONTINUE
         lth=j-ipdig
         if (lth.gt.0) jops=jops+ 8*lth
          y1(j)=  (a(iip1-j)-sum1)
          y2(j)=  (a(iip2-j)-sum2)
          a(iip1-j) = y1(j)/a(jp1)
          a(iip2-j) =  y2(j)/a(jp1)
           y1(j+1)= (a(iip1-j-1)-sum3-y1(j)*a(jjp2-j))
          y2(j+1) = (a(iip2-j-1)-sum4-y2(j)*a(jjp2-j))
          a(iip1-j-1)= y1(j+1)/a(jp2)
          a(iip2-j-1) = y2(j+1)/a(jp2)
         jops=jops + 12
 30    CONTINUE
          IF (IIP4 .LT. I-1) THEN
            IPLOCA=IIP4+1
            IIP4=I-1
        Copy X(IIP4) into TEMP
            go to 25
          ENDIF
            sum1=0.0
            sum2=0.0
            sum3=0.0

          DO 50 K=IpLOC,I-1
            sum1=sum1+a(iip1-k)*y1(k)
            sum2=sum2+a(iip1-k)*y2(k)
            sum3=sum3+a(iip2-k)*y2(k)
 50       CONTINUE
          jops = jops + 6*(i-iploc)
           a(ip1)=(a(ip1)-sum1)
           Produce X(i)=a(ip1)
          a(ip2+1)=(a(ip2+1)-sum2)/a(ip1)
          a(ip2) = (a(ip2) -sum3 -a(ip2+1)*a(ip2+1)*a(ip1))
          k=i+1
          Produce X(K)=a(ip2)
          jops=jops + 8
 20    End  Presched do

       ELSE
       jops=0
       Barrier
       jops=jops+1
        is1=neq-2*(neq/2)
        if (is1.eq.0) then
         is1=2
          if (maxa(3).eq.4) then
           B(2)=(b(2)-a(3)*b(1))
           jops=jops+3
          endif
        endif
       DO 510 I=is1+1,neq,2
       SUM=0.
        sum1=0.0
       JJ=MAXA(I)
       II=ICOLH(I)
```

```
         jp1=maxa(i+1)+1
        DO 520 J=II,1,-1
        SUM=SUM+A(JJ+J)*B(I-J)
         sum1=sum1+a(jp1+j)*b(i-j)
 520    CONTINUE
        jops=jops+ii*2+8
        B(I)=(B(I)-SUM)
        b(i+1)=(b(i+1) -sum1 -b(i)*a(jp1))
 510    continue
        do 1005 i=1,neq
        b(i)=b(i)/a(maxa(i))
1005    continue
        DO 1010 I=NEQ,is1+1,-2
         JJ= MAXA(I)
         jp1=maxa(i-1)
         B(I)=B(I)
         B(I-1)=(b(i-1) -a(jj+1)*b(i))
         lth=icolh(i)-1
c$dir no_recurrence
        DO 1020 J=I-ICOLH(I),I-2
        B(J)=b(J)-B(I)*A(JJ+I-J)-b(i-1)*a(jp1+i-j-1)
 1020   CONTINUE
        if (lth.gt.0) jops=jops+lth*4+4
 1010   Continue
        jops=jops+1
         if (is1.eq.2) then
           if (maxa(3) .eq. 4) then
            b(1)=(b(1)-a(3)*b(2))
           jops=jops+3
           else
            jops=jops+1
           endif
          endif
c......OUTPUT FROM LINEAR SOLVER
C*****C  WRITE(6,78) (B(I),I=1,6)
  78      FORMAT(2X,' SOLVER=',6E11.4)
        End Barrier
        ENDIF
        RETURN
        END
C%%%%%%%%%%%%%%%%%%%%%%%%%%%%
        SUBROUTINE ADD1(NEQ,ICOLH,maxa,nterms)
        INTEGER ICOLH(1),maxa(1)
        ISKIP=1
        IF(NEQ-2*(NEQ/2).EQ.0) ISKIP=2
        DO 201 J=ISKIP+1,NEQ,2
        IDIF=ICOLH(J+1)-ICOLH(J)
        IF(IDIF.LT.1) THEN
        ICOLH(J+1)=ICOLH(J)+1
        ELSE
        IF(IDIF.GT.1) THEN
        ICOLH(J)=ICOLH(J+1)-1
        ENDIF
        ENDIF
 201    CONTINUE
        do 20 i=1,neq+1
 20     maxa(i)=0
        maxa(1)=1
        maxa(2)=2
        do 10 i=2,neq
 10     maxa(i+1)=maxa(i)+icolh(i)+1
        nterms=maxa(neq+1)-1
        RETURN
        END
```

```
      Forcesub SOR1(N,NTERMS,A,B,X,C,CC,MAXA,NBW,TOL,MAXNIT,W,ICOLH)
     &  of NP ident ME
       DOUBLE PRECISION A(1),B(1),X(1),CC(1),TOL,W
       DOUBLE PRECISION C(1)
      Shared  DOUBLE PRECISION  THEMAX,THENOR
      INTEGER MAXA(1),ICOLH(1),N,ISOLVE,NBW,MAXNIT
      Shared LOGICAL TYPE3
      Shared LOGICAL TYPE4
      Shared LOGICAL TYPE1
      Shared LOGICAL TYPE2
      Shared INTEGER MSTAGL,MENDGL,IGO,NROL,ISKIP
      Private INTEGER MSTART,MEND
      Private  DOUBLE PRECISION TEMPP,SUM1,SUM2,XTEMP,TEMP
      End declarations
C     Barrier
C     write(6,*) 'first thing in SOR'
C     WRITE(6,*) 'MAXA(I)= ', (MAXA(I),I=1,N+1)
C     WRITE(6,*) 'ICOLH(I)= ', (ICOLH(I),I=1,N)
C     WRITE(6,*) 'A(NTERMS)= ', (A(I),I=1,MAXA(N+1)-1)
C     WRITE(6,*) 'B(NEQ)= ', (B(I),I=1,N)
C     End barrier
      ISKIP = 1
      IF ( N-2 * (N/2) .EQ. 0 ) ISKIP = 2

      DO 100 ICONT = 1,MAXNIT

c     Barrier
c     End barrier

      Presched do 11 JCONT = 1,NP
      DO 10 I = 1,N
      C(I) =  0.0
10    CONTINUE
11    End presched do

      Presched do 12 I = 1,N
      CC(I) = 0.0
12    End presched do
      Barrier
      IF (ISKIP .EQ. 2 ) THEN
         IF (ICOLH(2) .EQ. 1 ) THEN
         C(1) = X(2) * A(3)
         ENDIF
       ENDIF
      End barrier
C ****************   P R E S C H E D   D O   L O O P   ***************
C     Presched do 30 I = ISKIP+1,NROL,2
      Presched do 30 I = ISKIP+1 , N, 2
      C(I) = C(I) + X(I+1) * A(MAXA(I+1)+1)
      DO 20 J = I - ICOLH(I) ,I-1
C     C(J)=C(J) + X(I)*A(MAXA(I)+I-J)
      C(J)=C(J) + X(I)*A(MAXA(I)+I-J) + X(I+1) * A(MAXA(I+1)+I+1-J)
20    CONTINUE
30    End presched do


      Critical TYPE1
      XTEMP = 0.0
      TEMPP = 0.0
      DO 29 I = 1,N
      CC(I) = C(I) + CC(I)
29    CONTINUE
      End critical


      Barrier
```

```
      TEMP = X(1)
      X(1) =W*((B(1) - CC(1))/A(MAXA(1)))  + (1-W) * X(1)
C     THEMAX = ABS(TEMP - X(1))
      XTEMP = ((TEMP - X(1))**2)
      TEMPP = X(1)**2
      THEMAX = 0.0
      THENOR = 0.0
      End barrier

      Presched do 50 K = 2,N
      C(K) = B(K) -  CC(K)
      DO 40 J = K - ICOLH(K) ,K-1
40    C(K)  = C(K)  - A(MAXA(K) + K - J) * X(J)
      TEMP = X(K)
      X(K) = W*( C(K)  / A(MAXA(K))) + (1 - W ) * X(K)
C     TEMPP = ABS (X(K) - TEMP )
C     XTEMP = MAX ( TEMPP,XTEMP )
      XTEMP = XTEMP + ((X(K) - TEMP)**2)
      TEMPP = TEMPP + (X(K)**2)
50    End presched do

      Critical TYPE2
      THEMAX = THEMAX + XTEMP
      THENOR = THENOR + TEMPP
      End critical

      Barrier

      THEMAX = SQRT (THEMAX)
C     THEMAX = SQRT (THEMAX) / SQRT(THENOR)
      End barrier

C     write(6,*)' themax tol   ',themax,tol
      IF ( THEMAX .LT. TOL ) GO TO 110
100   CONTINUE
110   CONTINUE
      Barrier
      WRITE(6,*)' NUMBER OF ITTERATIONS IN GSM = ',ICONT
C     DO 79 I = 1 , 6
C     WRITE(6,78) X(1),X(2),X(3),X(4),X(5),X(6)
 78   FORMAT(2X,' S.O.R =',6E11.4)
      End Barrier
      RETURN
      END
```

APPENDIX D: SAP-4 Manual

REPORT NO.
EERC 73-11
JUNE 1973
REVISED APRIL 1974

EARTHQUAKE ENGINEERING RESEARCH CENTER

SAP IV

A STRUCTURAL ANALYSIS PROGRAM FOR STATIC AND DYNAMIC RESPONSE OF LINEAR SYSTEMS

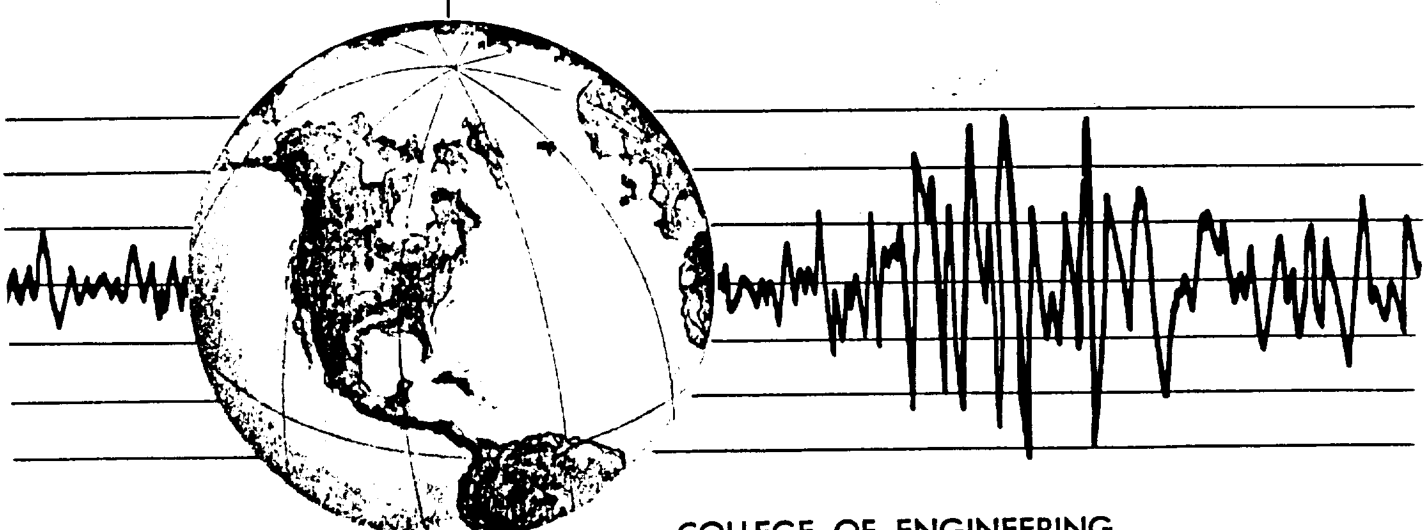
by

KLAUS-JÜRGEN BATHE

EDWARD L. WILSON

FRED E. PETERSON

A Report to the
National Science Foundation



COLLEGE OF ENGINEERING
UNIVERSITY OF CALIFORNIA · Berkeley, California

ABSTRACT

The computer program SAP IV for the static and dynamic analysis of linear structural systems is presented.

The report is divided into three parts. In the first part the reader is introduced to the logical construction of the program, the dynamic high speed storage allocation, the analysis capabilities, the finite element library and the numerical techniques used. Typical running times are given. In the second part of the report several sample analyses are described. These problems have been selected as standard problems whose solutions are provided with the program. In the last part of the report the user's manual of the program is given.

ACKNOWLEDGEMENTS

The development of the computer programs SAP including SAP IV has been supported by many organizations during the past years. The final phase of development and documentation of SAP IV was sponsored by Grants GI 36387 and GK 31586 from the National Science Foundation.

The release of the previous version of the program, SAP III, was restricted to agencies which sponsored our research. We are pleased that many institutions in Europe and the United States responded positively and that today we can make the latest version of the program available for duplication and mailing costs only. By making our work freely available, we hope that all those interested may profit from the developments that have taken place.

We would like to thank the following agencies, and in particular Engineering/Analysis Corporation, Berkeley, for their contributions towards the development of this program:

France

Informatique Internationale, Rungis

West-Germany

Germanischer Lloyd, Hamburg; Hochtief, Essen; Interatom, Bensberg/Köln; Kraftwerk Union, Erlangen; MAN, München

United States

Agbabian and Associates, Los Angeles, Calif.; Bechtel Corporation, San Francisco, Calif.; Beloit Corporation, Beloit, Wisconsin; Byron Jackson Pump Division of Borg Warner, Los Angeles, Calif.; Dames and Moore, San Francisco, Calif.; Engineering Mechanics Research Corporation, Troy, Michigan; Fluor Corporation, Los Angeles, Calif.; General Electric Company, San Jose, Calif.; Harza Engineering, Chicago, Illinois; International Harvester Company, Chicago, Illinois;

United States (continued)

Lockheed Missile and Space Company, Sunnyvale, Calif.; Martin and Associates, Los Angeles, Calif.; Philadelphia Gear Corporation, King of Prussia, Pennsylvania; Pregnoff/Matheu/Beebe, San Francisco, Calif.; Sargent and Lundy Engineers, Chicago, Illinois; Stone and Webster Engineering Corporation, Boston, Massachusetts; United Engineering, Philadelphia, Pennsylvania; U.S. Army Corps of Engineers - Waterways Experiment Station, Vicksburg, Mississippi; U.S. Army Corps of Engineers - Walla Walla District, Washington, D.C.; U.S. Department of the Interior, Bureau of Mines, Denver, Colorado; U.S. Naval Civil Engineering Laboratory, Port Hueneme, Calif.; Westinghouse Electric Corporation, Pittsburgh, Pennsylvania; Woodward-McNeill and Associates, Orange, Calif.; Yee and Associates, Honolulu, Hawaii.

TABLE OF CONTENTS

- PART A -

DESCRIPTION OF SAP IV

TABLE OF CONTENTS (Cont.)

- PART B -

SAMPLE ANALYSES

TABLE OF CONTENTS (Cont.)

- PART C -

APPENDICES

TABLE OF CONTENTS (Cont.)

- PART A -

DESCRIPTION OF SAP IV

1. INTRODUCTION

The development of an effective computer program for structural analysis requires a knowledge of three scientific disciplines -- structural mechanics, numerical analysis and computer application. The development of accurate and efficient structural elements requires a modern background in structural mechanics. The efficiency of a program depends largely on the numerical techniques employed and on their effective computer implementation. With regard to programming techniques, an optimum allocation of high and low speed storage is necessary.

A most important aspect of a general purpose computer program is, however, the ease with which it can be modified, extended and updated; otherwise, it may very well be that the program is obsolete within a few years after completion. This is because new structural elements are developed, better numerical procedures are available, or new computer equipment which requires new coding techniques is produced.

The structural analysis program SAP was designed to be modified and extended by the user. Additional options and new elements may easily be added. The program has the capacity to analyze very large three-dimensional systems; however, there is no loss in efficiency in the solution of smaller problems. Also, from the complete program, smaller special purpose programs can easily be assembled by simply using only those subroutines which are actually needed in the execution. This makes the program particularly usable on small size computers.

The current program version SAP IV for the static and dynamic analysis of linear structural systems is the result of several years' research and development experience. The program has proven to be a very flexible and efficient analysis tool. The program is coded in FORTRAN IV and operates without modifications on the CDC 6400, 6600 and 7600 computers. The first version of program SAP was published in September 1970 [28]. An improved static analysis program, namely SOLID SAP, or SAP II, was presented in 1971 [29]. Work was then started on a new static and dynamic analysis program. The program SAP III for static and dynamic analysis was released towards the end of 1972, but only to those agencies which supported our research. In relation to SAP III, the current version SAP IV has improvements throughout, and in particular has available a new variable-number-nodes thick shell and three-dimensional element, and out-of-core direct integration for time history analysis.

The structural systems to be analyzed may be composed of combinations of a number of different structural elements. The program presently contains the following element types:

(a) three-dimensional truss element,

(b) three-dimensional beam element,

(c) plane stress and plane strain element,

(d) two-dimensional axisymmetric solid,

(e) three-dimensional solid,

(f) variable-number-nodes thick shell and three-dimensional element,

(g) thin plate or thin shell element,

(h) boundary element,

(i) pipe element (tangent and bend).

These structural elements can be used in a static or dynamic analysis. The capacity of the program depends mainly on the total number of nodal points in the system, the number of eigenvalues needed in the dynamic analysis and the computer used. There is practically no restriction on the number of elements used, the number of load cases or the order and bandwidth of the stiffness matrix. Each nodal point in the system can have from zero to six displacement degrees of freedom. The element stiffness and mass matrices are assembled in condensed form; therefore, the program is equally efficient in the analysis of one-, two- or three-dimensional systems.

The formation of the structure matrices is carried out in the same way in a static or dynamic analysis. The static analysis is continued by solving the equations of equilibrium followed by the computation of element stresses. In a dynamic analysis the choice is between

1. frequency calculations only,
2. frequency calculations followed by response history analysis,
3. frequency calculations followed by response spectrum analysis,
4. response history analysis by direct integration.

To obtain the frequencies and vibration mode shapes solution routines are used which calculate the required eigenvalues and eigenvectors directly without a transformation of the structure stiffness matrix and mass matrix to a reduced form. In the direct integration an unconditionally stable integration scheme is used, which also operates on the original structure stiffness matrix and mass matrix. This way the program operation and necessary input data for a dynamic analysis is a simple addition to what is needed for a static analysis.

The purpose in this part of the report is to present briefly the general program organization, the current element library and the numerical techniques used. The different options available for static and dynamic analyses are described and typical running times are given. In the presentation, emphasis is directed to the practical aspects of the program. For information on the development of the structural elements and the numerical techniques used the reader is referred to appropriate references.

2. THE EQUILIBRIUM EQUATIONS FOR COMPLEX STRUCTURAL SYSTEMS

2.1 Element to Structure Matrices

The nodal point equilibrium equations for a linear system of structural elements can be derived by several different approaches [1] [2] [9] [15] [23] [34]. All methods yield a set of linear equations of the following form

$$M\ddot{u} + C\dot{u} + Ku = R \tag{1}$$

where M is the mass matrix, C is the damping matrix and K is the stiffness matrix of the element assemblage; the vectors u, $\dot{u}$, $\ddot{u}$ and R are the nodal displacements, velocities, accelerations and generalized loads, respectively. The structure matrices are formed by direct addition of the element matrices; for example

$$K = \sum K_m \tag{2}$$

where K_m is the stiffness matrix of the m'th element. Although K_m is formally of the same order as K, only those terms in K_m which pertain to the element degrees of freedom are nonzero. The addition of the element matrices can therefore be performed by using the element matrices in compact form together with identification arrays which relate element to structure degrees of freedom. The algorithm used in the program is described in Section 3.3.

In the program the structure stiffness matrix and a diagonal mass matrix are assembled. Therefore, a lumped mass analysis is assumed, where the structure mass is the sum of the individual element mass matrices plus additional concentrated masses which are specified at selected degrees of freedom. The damping is assumed to be proportional and is specified in form of a modal damping factor. The assumptions used in lumped mass analyses and in the use of proportional damping have been discussed at various occasions [9] [11] [17] [33].

2.2 Boundary Conditions

If a displacement component is zero, the corresponding equation is not retained in the structure equilibrium equations, Eq. (1), and the corresponding element stiffness and mass terms are disregarded. If a non-zero displacement is to be specified at a degree of freedom i, say $u_i = x$, the equation

$$k\,u_i = k\,x \tag{3}$$

is added into Eq. (1), where $k \gg k_{ii}$. Therefore, the solution of Eq. (1) must give $u_i = x$. Physically, this can be interpreted as adding at the degree of freedom "i" a spring of large stiffness k and specifying a load which, because of the relatively flexible structure at this degree of freedom, produces the required displacement x.

3. PROGRAM ORGANIZATION FOR CALCULATION OF THE STRUCTURE STIFFNESS MATRIX AND MASS MATRIX

The calculation of the structure stiffness matrix and mass matrix is accomplished in three distinct phases:

1. The nodal point input data is read and generated by the program. In this phase the equation numbers for the active degrees of freedom at each nodal point are established.
2. The element stiffness and mass matrices are calculated together with their connection arrays; the arrays are stored in sequence on tape (or other low-speed storage).
3. The structure stiffness matrix and mass matrix are formed by addition of the element matrices and stored in block form on tape.

It need be noted that these basic steps are indenendent of the element type used and are the same for either a static or dynamic analysis.

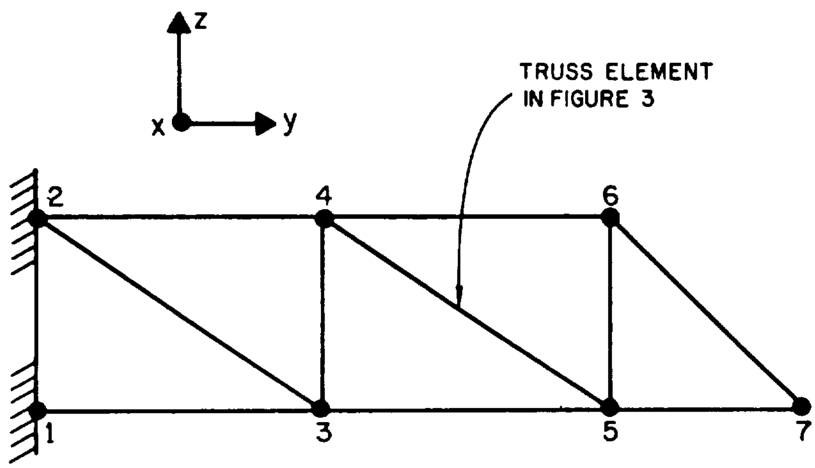
3.1 Nodal Point Input Data and Degrees of Freedom

The capacity of the program is controlled by the number of nodal points of the structural system. For each nodal point six boundary condition codes (stored in the array ID), three coordinates (stored in the arrays X,Y,Z) and the nodal point temperatures (stored in the array T) are required (generation capability is provided). All nodal point data is retained in high speed storage during the formation of the element stiffness and mass matrices. Since the required high speed storage for the element subroutines is relatively small, the minimum required storage for a given problem is a little larger than ten times the number of nodal points in the system.

It need be noted that the user should allow only those degrees of freedom which are compatible with the elements connected to a nodal point. The program always deals with six possible degrees of freedom at each nodal point, and all non-active degrees of freedom should be deleted, so as to decrease the order of the structure matrices. Specifically, a "1" in the ID array denotes that no equation shall be associated with the degree of freedom, whereas a "0" indicates that this is an active degree of freedom. Figure 1 shows for the simple truss structure the ID array as it was read and/or generated by the program. Once the complete ID and X,Y,Z arrays have been obtained, equation numbers are associated with all active degrees of freedom, i.e., the zeroes in the ID array are replaced by corresponding equation numbers, and each one is replaced by a zero, as shown in Fig. 2 for the simple truss example.

3.2 Element Mass and Stiffness Calculations

With the coordinates of all nodal points known and the equation numbers of the degrees of freedom having been established, the stiffness, mass and stress-displacement transformation matrices for each structural element in the system are calculated. As pointed out earlier, little additional high-speed storage is required for this phase since these matrices are formed and placed on tape storage at the same time as the element properties are read. Together with the matrices pertaining to the element, the corresponding element connection array, vector LM, is written on tape. The vector LM is established



NODAL POINT LAYOUT OF TRUSS

$$
\text{ID} = \begin{array}{c}
\\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7
\end{array}
\begin{array}{c}
\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \end{array} \\
\left[\begin{array}{cccccc}
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 1 & 1 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1 \\
1 & 0 & 0 & 1 & 1 & 1
\end{array}\right]
\end{array}
$$

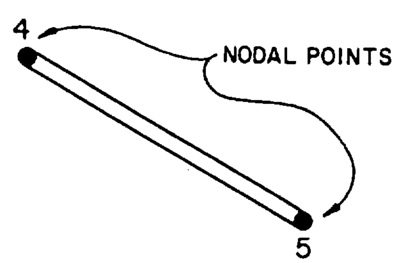
DEGREES OF FREEDOM

NODAL POINT NUMBERS

FIGURE 1: NODAL POINT LAYOUT OF TRUSS–EXAMPLE AND ID–ARRAY AS READ AND/OR GENERATED

$$
ID = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 2 & 0 & 0 & 0 \\
0 & 3 & 4 & 0 & 0 & 0 \\
0 & 5 & 6 & 0 & 0 & 0 \\
0 & 7 & 8 & 0 & 0 & 0 \\
0 & 9 & 10 & 0 & 0 & 0
\end{bmatrix}
$$

FIGURE 2: ID ARRAY OF TRUSS–EXAMPLE AFTER ALLOCATION OF EQUATION NUMBERS TO ACTIVE DEGREES OF FREEDOM



$$
LM = \begin{bmatrix} 0 \\ 3 \\ 4 \\ 0 \\ 5 \\ 6 \end{bmatrix}
$$

FIGURE 3: CONNECTION ARRAY (VECTOR LM) FOR A TYPICAL ELEMENT OF THE TRUSS–EXAMPLE

from the ID matrix and the specified structure nodal points pertaining to the element. The connection array for a typical element of the truss element is shown in Fig. 3.

The element matrices are calculated in groups, i.e., always all elements in one group together, thus calling the corresponding element subroutine only once for each element group. After all element matrices have been established, the ID and X,Y,Z arrays are not needed any more, and the corresponding storage area is used for the formation of the structure matrices and later for the solution of the equations of equilibrium.

3.3 Formation of Structure Stiffness and Mass

The stiffness matrix and mass matrix of the structure are formed in blocks, as shown in Fig. 4 for the truss-example. The number of equations per block depends on the available high speed storage and is calculated in the program as indicated in Fig. 5. It is noted that on reasonable size computers very large systems can be analyzed for static and dynamic response. With the number of equations per block known, the stiffness and mass matrix are assembled two blocks at a time by direct addition of the element matrices. In this process it is necessary to pass through the element matrices which are stored on tape. In order to minimize tape reading, in each pass element matrices which pertain to the next several blocks are written on

To avoid rewind and then read again

another tape. This way the tape reading necessary for the formation of these blocks is reduced significantly.

A flow diagram of the program organization for the calculation of the structure stiffness matrix and mass matrix is shown in Fig. 6.

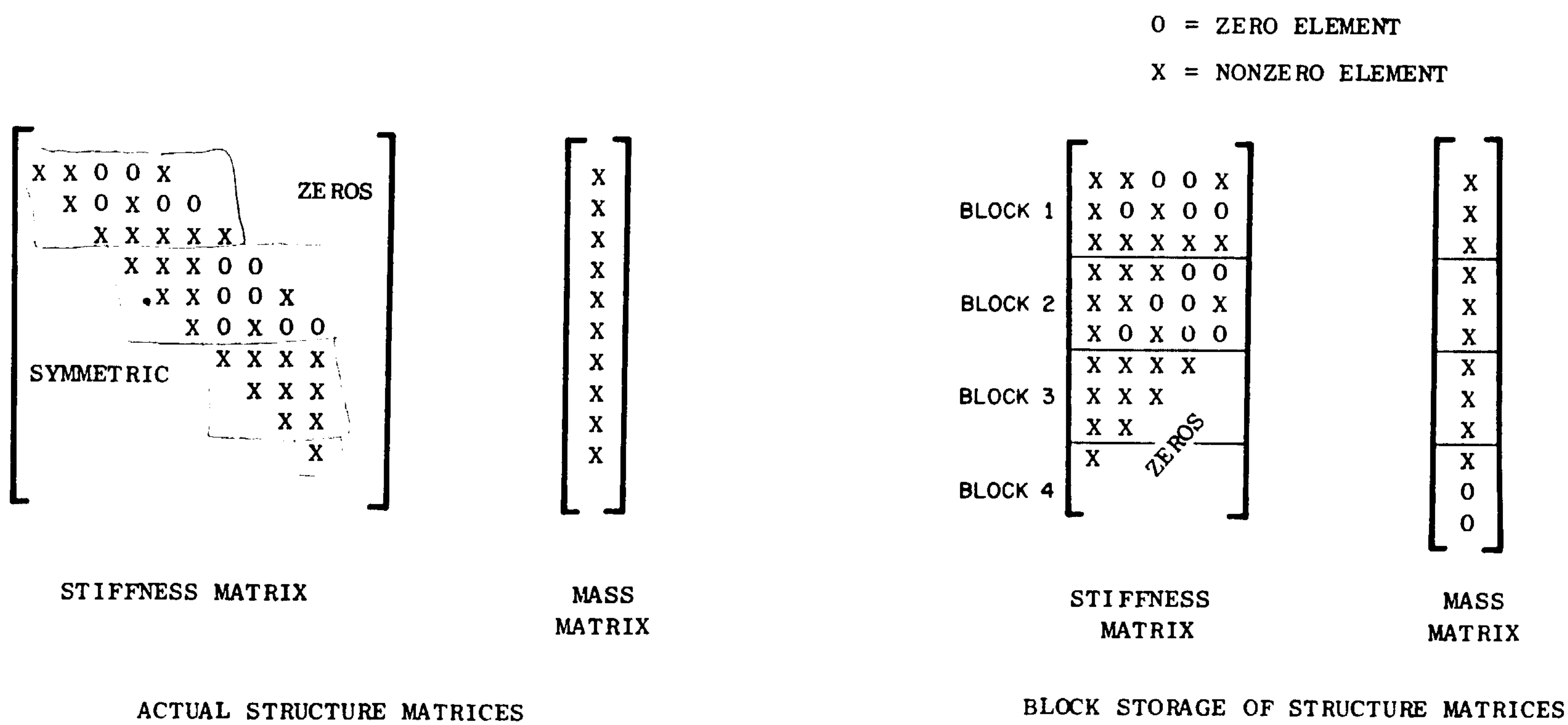


FIGURE 4: STORAGE OF STIFFNESS MATRIX AND MASS MATRIX ON TAPE

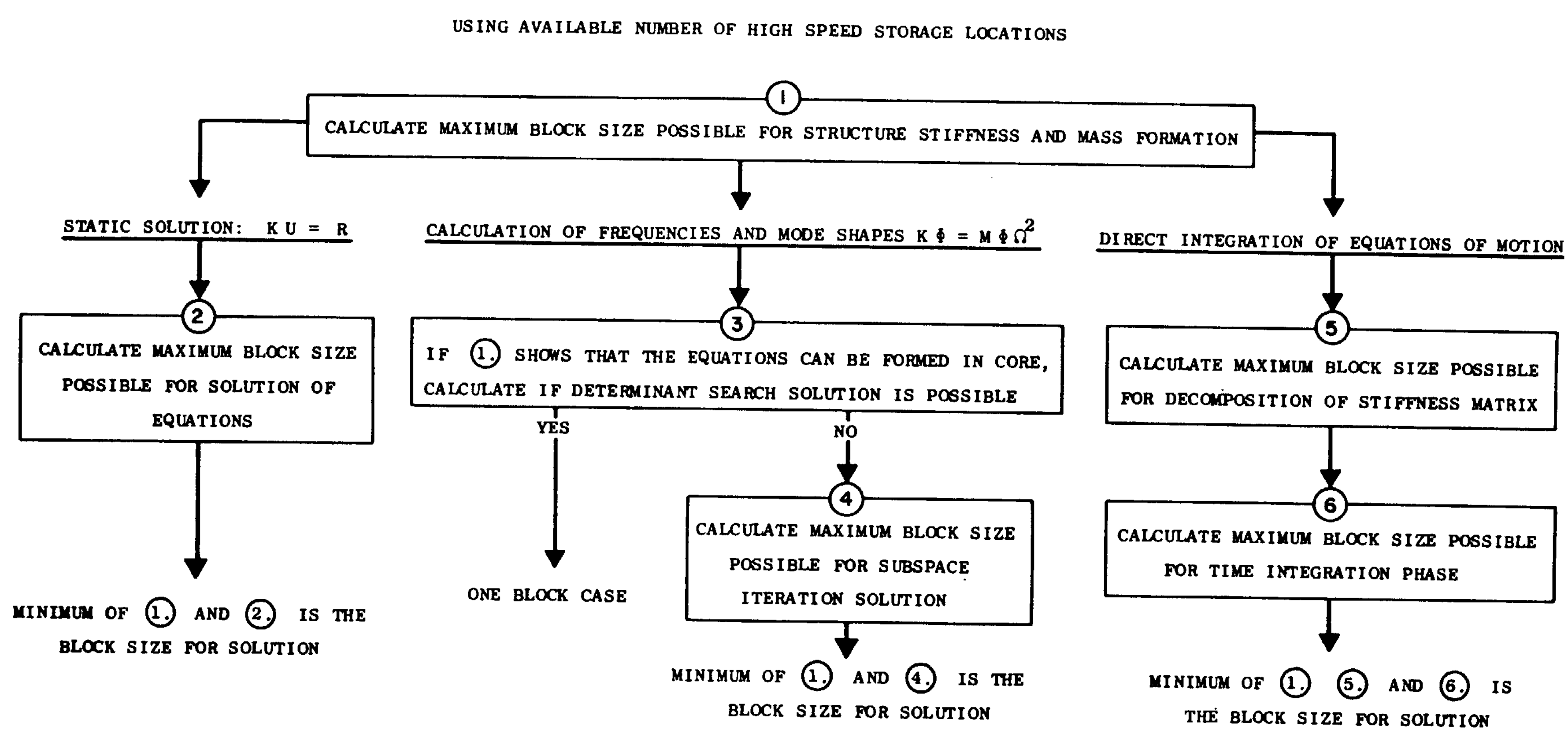


FIGURE 5: FLOWCHART SHOWING CALCULATION OF NUMBER OF EQUATIONS IN A BLOCK

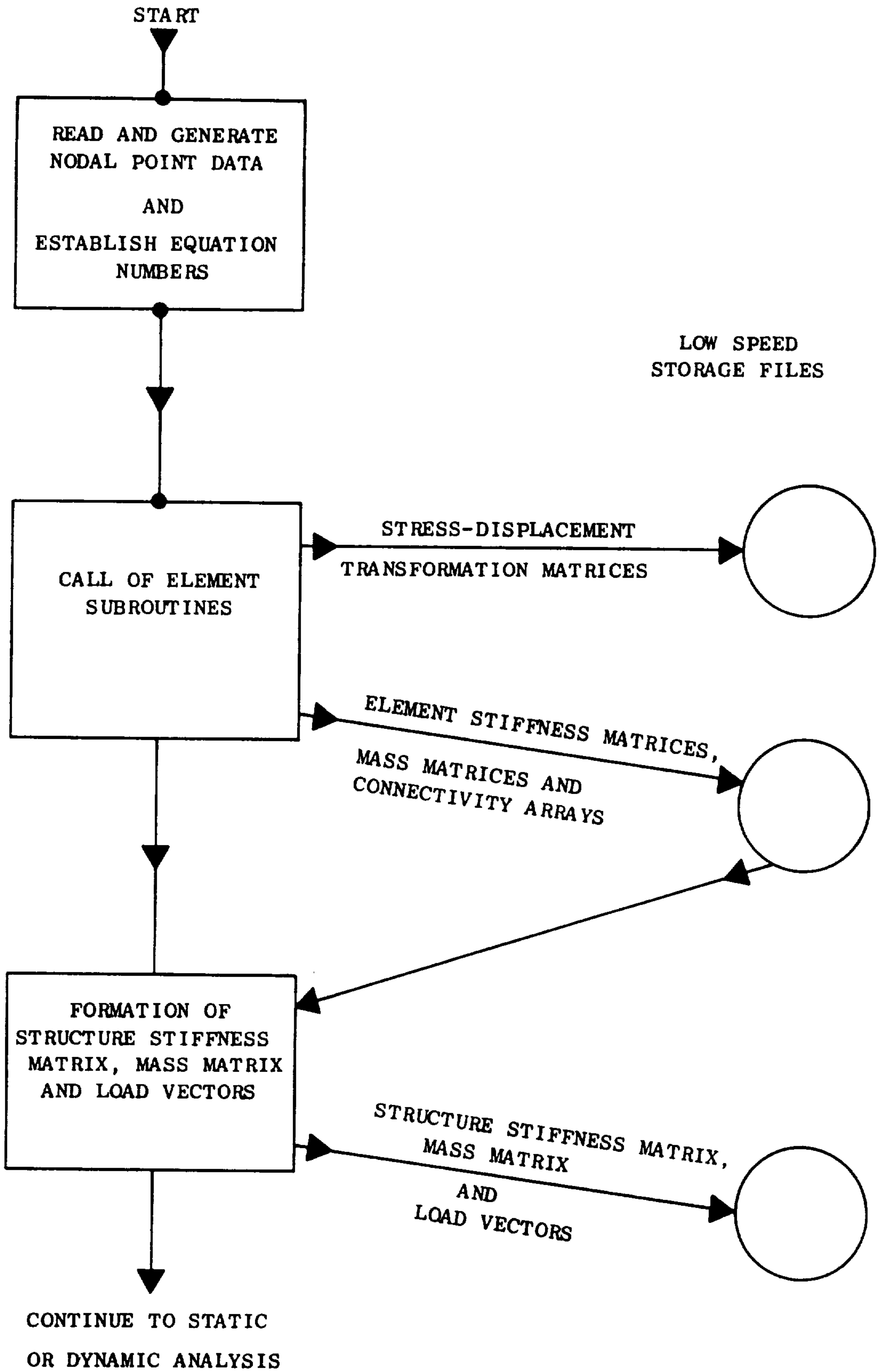


FIGURE 6: FLOWCHART FOR CALCULATION OF STRUCTURE STIFFNESS MATRIX AND MASS MATRIX

With the matrices stored in block form on tape either a static or a dynamic analysis can now be carried out.

4. THE ELEMENT LIBRARY

The element library of SAP IV consists of eight different element types. These elements can be used in either a static or dynamic analysis. They are shown in Fig. 7 and are briefly described below.

4.1 Three Dimensional Truss Element

The derivation of the truss element stiffness is given in Refs. [23] [29]. The element can be subjected to a uniform temperature change.

4.2 Three-Dimensional Beam Element

The beam element included in the program considers torsion, bending about two axes, axial and shearing deformations. The element is prismatic. The development of its stiffness properties is standard and is given in Ref. [23]. Inertia loading in three directions and specified fixed-end-forces form the element load cases. Forces (axial and shear) and moments (bending and torsion) are calculated in the beam local co-ordinate system.

A typical beam element is shown in Fig. 7b. A plane which defines the principal bending axis of the beam is specified by the plane i, j, k. Only the geometry of nodal point k is needed; therefore, no additional degrees of freedom for nodal point k are used in the computer program. A unique option of the beam member is that the ends of the beam can be geometrically constrained to a master node. Slave degrees of freedom at the end of the beam are eliminated from the formulation and replaced by the transformed degrees of freedom of the master node [18] [29]. This technique reduces the total number of joint equilibrium

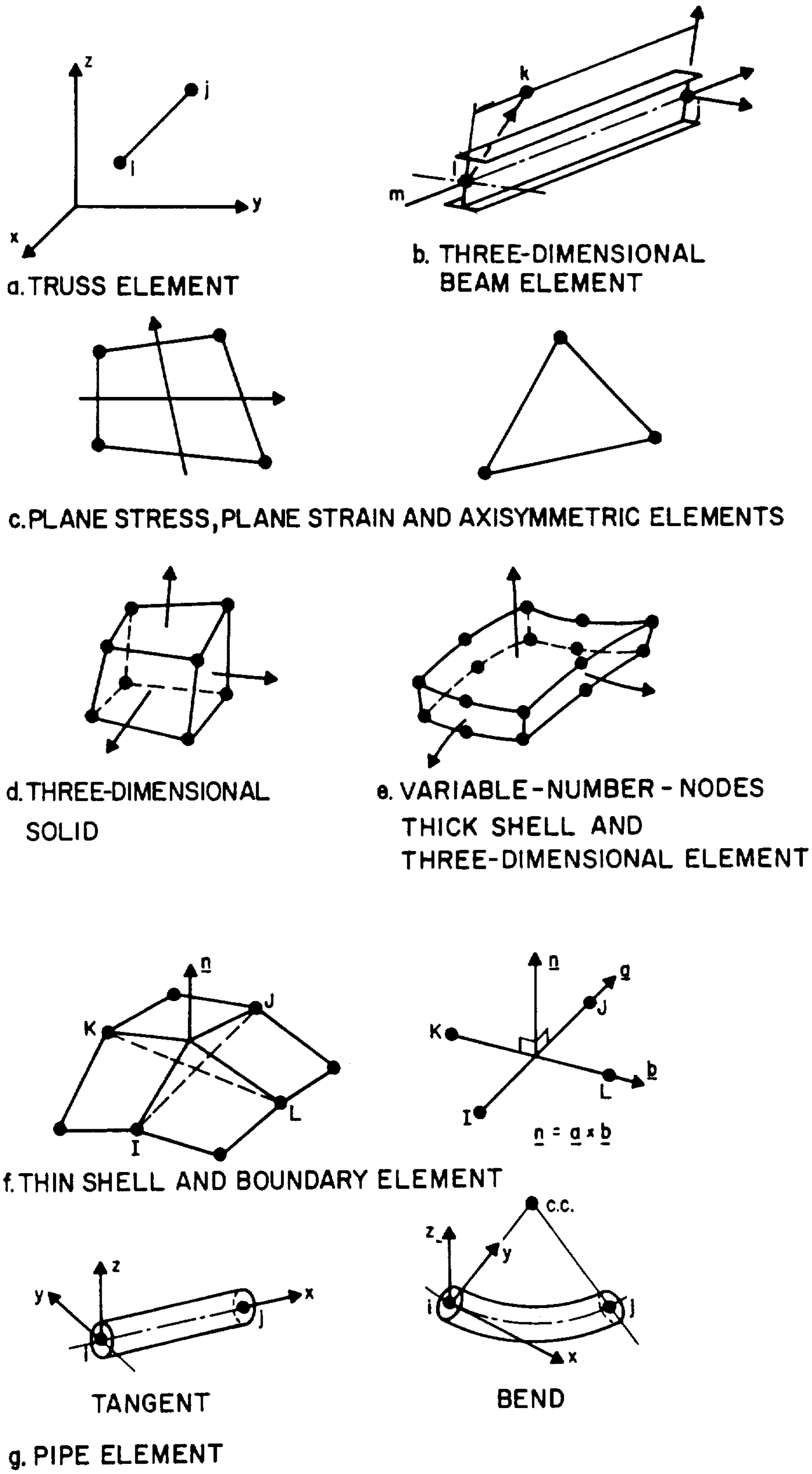


FIGURE 7: ELEMENT LIBRARY OF SAP IV

equations in the system (while possibly increasing the bandwidth) and greatly reduces the possibility of numerical sensitivities in many types of structures. Also, the method can be used to specify rigid floor diaphragms in building analysis.

4.3 Plane Stress, Plane Strain and Axisymmetric Elements

A plane stress quadrilateral (or triangular) element with orthotropic material properties is available. Each plane stress element may be of different thickness and may be located in an arbitrary plane with respect to the three-dimensional coordinate system. The plane strain and axisymmetric elements are restricted to the y-z plane. Gravity, inertia and temperature loadings may be considered. Stresses may be computed at the center of the element and at the center of each side. The element is based on an isoparametric formulation [19] [34]. Incompatible displacement modes can be included in order to improve the bending properties of the element [26] [29] [32].

4.4 Three-Dimensional Solid Element

A general eight nodal point "brick" element, with three translational degrees of freedom per nodal point can be used, Fig. 7d. Isotropic material properties are assumed and element loading consists of temperature, surface pressure and inertia loads in three directions. Stresses (six components) may be computed at the center of the element and at the center of each face. The element employs incompatible modes, which can be very effective if rectangular elements are used [26].

4.5 Variable-Number-Nodes Thick Shell and Three-Dimensional Element

A general three-dimensional isoparametric or subparametric element which may have from 8 to 21 nodes can be used for three-dimensional or thick shell analysis, Fig. 7e [7] [8]. General orthotropic material properties can be assigned to the element. The loading may consist of applied surface pressure, hydrostatic loads, inertia loads in three directions, and thermal loads. Six global stresses are output at up to seven locations within an element.

4.6 Thin Plate and Shell Element

The thin shell element available in the program is a quadrilateral of arbitrary geometry formed from four compatible triangles. The bending and plane stress properties of the element are described in references [12] [14]. The shell element uses the constant strain triangle and the LCCT9 element to represent the membrane and bending behavior, respectively. The central node is located at the average of the coordinates of the four corner nodes. The element has six interior degrees of freedom which are eliminated at the element level prior to assembly; therefore, the resulting quadrilateral element has twenty-four degrees of freedom, i.e., six degrees of freedom per node in the global coordinate system.

In the analysis of flat plates the stiffness associated with the rotation normal to the shell surface is not defined; therefore, the rotation normal degree of freedom must not be included in the analysis. For curved shells, the normal rotation need be included as an extra degree of freedom. In case the curvature is very small, the degree of freedom should be restrained by the addition of a "Boundary Element" with a small normal rotational stiffness, say of less or about 10% of the element bending stiffness [13] [34].

4.7 Boundary Element

The boundary element, shown in Fig. 7f, can be used for the following:

1. in the idealization of an external elastic support at a node;
2. in the idealization of an inclined roller support;
3. to specify a displacement, or
4. to eliminate the numerical difficulty associated with the 'sixth' degree of freedom in the analysis of nearly flat shells.

The element is one-dimensional with an axial or torsional stiffness. The element stiffness coefficients are added directly to the total stiffness matrix (see Section 2.2).

4.8 Pipe Element

The pipe element (Fig. 7g) can represent a straight segment (tangent) or a circularly curved segment (bend); both elements require a uniform section and uniform material properties. Elements can be directed arbitrarily in space. The member stiffness matrices account for bending, torsional, axial and shearing deformations. In addition, the effect of internal pressure on the stiffness of curved pipe elements is considered.

The types of structure loads contributed by the pipe elements include gravity loading in the global directions, and loads due to thermal distortions and deformations induced by internal pressure. Forces and moments acting at the member ends (i,j) and at the center of each bend are calculated in coordinate systems aligned with the member's cross section.

The pipe element stiffness matrix is formed by first evaluating the flexibility matrix corresponding to the six degrees of freedom at end j as given by Poley [22]. With the corresponding stiffness matrix, the equilibrium transformations outlined by Hall et al [16] are used to form the complete element stiffness matrix. Distortions due to element loads are premultiplied by the stiffness matrix to compute restrained nodal forces due to thermal, pressure or gravity loads.

5. STATIC ANALYSIS

A static analysis involves the solution of the equilibrium equations

$$K u = R \tag{4}$$

followed by the calculation of element stresses.

5.1 Solution of Equilibrium Equations

The load vectors R have been assembled at the same time as the structure stiffness matrix and mass matrix were formed. The solution of the equations is obtained using the large capacity linear equation solver SESOL [31]. This subroutine uses Gauss elimination on the positive-definite symmetrical system of equations. The algorithm performs a minimum number of operations; i.e. there are no operations with zero elements. In the program, the L^TDL decomposition of K is used, hence Eq. (4) can be written as

$$L^T v = R \tag{5}$$

and

$$v = DLu \tag{6}$$

where the solution for v in Eq. (5) is obtained by a reduction of the load vectors; the displacement vectors u are then calculated by a back-substitution.

In the solution, the load vectors are reduced at the same time as K is decomposed. In all operations it is necessary to have at any one time the required matrix elements in high-speed storage. In the reduction, two blocks are in high speed storage (as was also the case in the formation of the stiffness matrix and mass matrix), i.e., the "leading" block, which finally stores the elements of L and D, and in succession those blocks which are affected by the decomposition of the "leading" block. Table 1 gives some typical solution times.

5.2 Evaluation of Element Stresses

After the nodal point displacements have been evaluated, sequentually the element stress-displacement matrices are read from low speed storage and the element stresses are calculated.

TABLE 1 SOLUTION OF EQUATIONS USING SESOL

NUMBER OF EQUATIONS	HALF BANDWIDTH	CENTRAL PROCESSOR SEC	COMPUTER USED
8036	544	1786 †	CDC 6600
2696	488	1260	CDC 6600
4214	205	31	CDC 7600

† The inner DO - loop in the factorization of the stiffness matrix has been coded in machine language for this solution.

6. CALCULATION OF FREQUENCIES AND MODE SHAPES

The dynamic analysis of a structural system using mode superposition requires as the first step the solution of the generalized eigenvalue problem

$$K \phi = \omega^2 M \phi \qquad (7)$$

where ω and ϕ are free vibration frequency and mode shape, respectively. As was described in Section 3.3 the program stores the stiffness and mass matrix in blocks on tape, Fig. 4. The mass matrix is diagonal with partly zero diagonal elements. The program assumes that only the lowest p eigenvalues and corresponding eigenvectors are needed. The solution of Eq. (7) can therefore be written as

$$K \Phi = M \Phi \Omega^2 \qquad (8)$$

where Ω^2 is a diagonal matrix with the p smallest eigenvalues, i.e. $\Omega^2 = \mathrm{diag}(\omega_i^2)$, and Φ stores the corresponding M-orthonormalized eigenvectors $\phi_1, \phi_2, \ldots \phi_p$. Two different solution procedures are used in the program, a determinant search technique or a subspace iteration solution. The determinant search solution is carried out when the stiffness matrix can be contained in high-speed storage in one block. Therefore, for systems of large order and bandwidth the subspace iteration method is used. Both solution techniques solve the generalized eigenvalue problem directly without a transformation to the standard form [3].

6.1 The Determinant Search Solution

The determinant search technique is best suited for the analysis of large systems in which K and M have small bandwidths [4]. Basically, the solution algorithm combines triangular factorization and vector inverse iteration in an optimum manner to calculate the required eigenvalues and eigenvectors; these are obtained in sequence starting from the least dominant eigenpair ω_1^2, Φ_1. An efficient accelerated secant iteration procedure which operates on the characteristic polynomial

$$p(\omega^2) = \det(K - \omega^2 M) \tag{9}$$

is used to obtain a shift near the next unknown eigenvalue. The eigenvalue separation theorem (Sturm sequence property) is used in this iteration. Each determinant evaluation requires a triangular factorization of the matrix $K - \omega^2 M$. Once a shift near the unknown eigenvalue has been obtained, inverse iteration is used to calculate the eigenvector; the eigenvalue is obtained by adding the Rayleigh quotient correction to the shift value. Table 2 shows typical solution times.

6.2 The Subspace Iteration Solution

When the system is too large to be completely contained in high speed storage, i.e. more blocks than one are used, the subspace iteration solution is carried out. The iteration can be interpreted as a repeated application of the Ritz method [5] [9], in which the computed eigenvectors from one step are used as the trial basis vectors for the next iteration until convergence to the required p eigenvalues and

TABLE 2 CALCULATION OF FREQUENCIES AND MODE SHAPES USING DETERMINANT SEARCH METHOD

SYSTEM	SYSTEM ORDER n	MAXIMUM HALF BAND WIDTH	NUMBER OF REQ'D. FREQN. AND MODE SHAPES p	COMPUTER USED	CENTRAL PROCESSOR SEC
PLANE FRAME	297	30	3	CDC 6400	40
PIPING SYSTEM	566	12	7	CDC 6600	11
BUILDING	340	32	7	CDC 6600	20
CONTAINER	265	65	40	CDC 7600	58

eigenvectors is obtained.

The solution is carried out by iterating simultaneously with q linearly independent vectors, where $q > p$. In the k'th iteration the vectors span the q-dimensional subspace $\mathcal{E}_k$ and 'best' eigenvalue and eigenvector approximations are calculated; i.e. when the vectors span the p-dimensional least dominant subspace, the required eigenvalues and eigenvectors are obtained.

Let V_o store the starting vectors, then the k'th iteration is described as follows:

Solve for vectors $\overline{V}_k$ which span $\mathcal{E}_k$

$$K \overline{V}_k = M V_{k-1} \tag{10}$$

Calculate the projections of K and M onto $\mathcal{E}_k$ (i.e. the generalized stiffness matrix and mass matrix corresponding to $\mathcal{E}_k$)

$$K_k = \overline{V}_k^T K \overline{V}_k \tag{11}$$

$$M_k = \overline{V}_k^T M \overline{V}_k \tag{12}$$

Solve for the eigensystem of K_k and M_k

$$K_k Q_k = M_k Q_k \Omega_k^2 \tag{13}$$

and calculate the k'th improved approximation to the eigenvectors

$$V_k = \overline{V}_k Q_k \tag{14}$$

Provided that the starting subspace is not orthogonal to any of the required eigenvectors, the iteration converges to the desired result, i.e. $\Omega_k^2 \rightarrow \Omega^2$ and $V_k \rightarrow \Phi$ as $k \rightarrow \infty$.

The number of vectors q used in the iteration is taken greater than the desired number of eigenvectors in order to accelerate the convergence of the process. The number of iterations required to achieve satisfactory convergence depends, of course, on the quality of the starting vectors V_o. Unless requested otherwise (see Section 6.3), the program generates q starting vectors where q = min(2p, p+8), which has proven to be effective in general applications. At convergence a Sturm sequence check can be requested to verify that the lowest p eigenvalues have been found.

Table 3 lists a few typical solution times using the program generated starting vectors.

6.3 Dynamic Optimization

The solution of the eigenvalue problem may be required when a good estimate of the required eigensystem is already known, such as in dynamic optimization. In this case the subspace iteration method is ideally suited for solution. The number of iteration vectors q and the vectors V_o together with the maximum number of iterations can in this case be specified by the user. Also, in case the number of eigenvalues and vectors required is increased, the already calculated eigenvectors can be specified as part of the starting iteration vectors in order to accelerate convergence.

TABLE 3 CALCULATION OF FREQUENCIES AND MODE SHAPES USING SUBSPACE ITERATION METHOD

SYSTEM	SYSTEM ORDER n	MAXIMUM HALF BAND WIDTH	NUMBER OF REQ'D. FREQN. AND MODE SHAPES p	COMPUTER USED	CENTRAL PROCESSOR SEC
PLANE FRAME	297	30	3	CDC 6400	25
PIPING SYSTEM	566	12	28	CDC 6600	142
BLDG. WITH FOUNDATION	1174	138	45	CDC 6600	890
3-DIM BLDG. FRAME	468	156	4	CDC 6400	160

7. DYNAMIC ANALYSES

In dynamic response analysis the solution of the equations

$$M\ddot{u} + C\dot{u} + Ku = R(t) \tag{15}$$

is required, where R(t) can be a vector of arbitrary time varying loads or of effective loads which result from ground motion. Specifically, in the case of ground motion, if it is assumed that the structure is uniformly subjected to the ground acceleration $\ddot{u}_g$ [9], the equilibrium equations considered are

$$M\ddot{u}_r + C\dot{u}_r + Ku_r = -M\ddot{u}_g \tag{16}$$

where u_r is the relative displacement of the structure with respect to the ground, i.e. $u_r = u - u_g$.

The program can carry out a history analysis for solution of Eqs. (15) or (16), or a response spectrum analysis for solution of Eq. (16). The history analysis can be carried out using mode superposition or direct integration. The response spectrum analysis necessitates, of course, first the solution of the required eigensystem.

7.1 Response History Analysis by Mode Superposition

In the mode superposition analysis, it is assumed that the structural response can be described adequately by the p lowest vibration modes, where $p \ll n$. Using the transformation $u = \Phi X$, where the columns in Φ are the p M-orthonormalized eigenvectors, Eq. (15) can be written as

$$\ddot{X} + \Delta\dot{X} + \Omega^2 X = \Phi^T R \tag{17}$$

where

$$\Delta = \mathrm{diag}(2\omega_i \xi_i)\,; \qquad \Omega^2 = \mathrm{diag}(\omega_i^2) \tag{18}$$

In Eq. (18) it is assumed that the damping matrix C satisfies the modal orthogonality condition

$$\phi_i^T C \phi_j = 0 \qquad (i \neq j) \tag{19}$$

Equation (17) therefore represents p uncoupled second order differential equations. These are solved in the program using the Wilson θ-method, which is an unconditionally stable step-by-step integration scheme [6]. The same time step is used in the integration of all equations to simplify the calculation of stress components at preselected times.

In the case of prescribed ground motion $u_r = \phi X$ and in Eq. (17) the right hand side is given by $-\phi^T M \ddot{u}_g$, where the ground acceleration is considered as the sum of the components in the x, y and z directions as described in Section 7.3.

7.2 Response History Analysis by Direct Integration

The solution of the equations of motion, Eqs. (15) and (16), can be obtained by direct integration [6]. In the program the Wilson θ-method is used, which is unconditionally stable. The algorithm employed is summarized in Table 4. It need be noted that Rayleigh damping is assumed, i.e. $C = \alpha M + \beta K$ [11]. This form of damping is easily taken account of in the analysis, because no storage and no multiplications for a damping matrix are required.

TABLE 4: STEP-BY-STEP DIRECT INTEGRATION ALGORITHM

Initial Calculations

1. Calculate the following constants (Assume $C = \alpha M + \beta K$).

$\theta = 1.4, \quad \tau = \theta\Delta t$

$a_o = (6 + 3\alpha\tau)/(\tau^2 + 3\beta\tau)$

$b_o = \alpha - \beta a_o$

$a_1 = 6/\tau^2 + 3b_o/\tau$

$a_2 = 6/\tau + 2b_o$

$a_3 = 2 + \tau b_o/2$

$a_4 = 6/[\theta(3\beta\tau + \tau^2)]$

$b_1 = \beta a_4$

$a_5 = 3b_1/\tau - 6/(\tau^2\theta)$

$a_6 = 2b_1 - 6/(\tau\theta)$

$a_7 = b_1\tau/2 + 1 - 3/\theta$

$a_8 = \Delta t/2$

$a_9 = \Delta t^2/3$

$a_{10} = \frac{1}{2} a_9$

2. Form effective stiffness matrix $K^* = K + a_o M$.

3. Triangularize K^*

For Each Time Increment

1. Form effective load vector R_t^*.

$$R_t^* = R_t + \theta(R_{t+\Delta t} - R_t) + M[a_1 u_t + a_2\dot{u}_t + a_3\ddot{u}_t].$$

2. Solve for effective displacement vector u_t^*.

$$K^* u_t^* = R_t^*$$

3. Calculate new acceleration, velocity and displacement vectors,

$$\ddot{u}_{t+\Delta t} = a_4 u_t^* + a_5 u_t + a_6\dot{u}_t + a_7\ddot{u}_t$$

$$\dot{u}_{t+\Delta t} = \dot{u}_t + a_8(\ddot{u}_{t+\Delta t} + \ddot{u}_t)$$

$$u_{t+\Delta t} = u_t + \Delta t\,\dot{u}_t + a_9\ddot{u}_t + a_{10}\ddot{u}_{t+\Delta t}$$

4. Calculate element stresses if desired.

7.3 Response Spectrum Analysis

In this analysis the ground acceleration vector in Eq. (16) is written as

$$\ddot{u}_g = \ddot{u}_{gx} + \ddot{u}_{gy} + \ddot{u}_{gz} \tag{20}$$

where $\ddot{u}_{gx}$, $\ddot{u}_{gy}$ and $\ddot{u}_{gz}$ are the ground accelerations in the x, y and z directions, respectively. The equation for the response in the r'th mode is therefore

$$\ddot{x}_r + 2\xi_r \omega_r \dot{x}_r + \omega_r^2 x_r = r_{rx} + r_{ry} + r_{rz} \tag{21}$$

where x_r is the r'th element in X and

$$r_{rx} = -\phi_r^T M \ddot{u}_{gx} \; ; \; r_{ry} = -\phi_r^T M \ddot{u}_{gy} \; ; \; r_{rz} = -\phi_r^T M \ddot{u}_{gz} \tag{22}$$

Using the definition of the spectral displacement [10], the maximum absolute modal displacements of the structure subjected to an acceleration into the x direction are

$$u_{rx}^{(max)} = \phi_r \left| \phi_r^T M I_x \right| S_x(\omega_r) \tag{23}$$

where $S_x(\omega_r)$ is the spectral displacement into the x direction corresponding to the frequency ω_r and I_x is a null vector except that those elements are equal to one which correspond to the x-translational degrees of freedom. Similarly, for the responses due to a ground acceleration into the y and z-directions

$$u_{ry}^{(max)} = \phi_r \left| \phi_r^T M I_y \right| S_y(\omega_r) \quad ; \quad u_{rz}^{(max)} = \phi_r \left| \phi_r^T M I_z \right| S_z(\omega_r) \tag{24}$$

and the total maximum response in the r'th mode is assumed to be

$$u_r^{(max)} = u_{rx}^{(max)} + u_{ry}^{(max)} + u_{rz}^{(max)} \tag{25}$$

Program SAP IV calculates the maximum responses in each of the p lowest modes, where the spectra (displacements or accelerations) into the x, y and z-directions are assumed to be proportional to each other. The total response for displacements and stress resultants is calculated as the square root of the sum of the squares of the modal maximum responses [10] [36].

7.4 Restart Capability in Mode Superposition Analysis

The most expensive phase in mode superposition analysis is usually the calculation of frequencies and mode shapes. However, once the required eigensystem has been solved for, it can be used to analyze the structure for different loading conditions. Also, in a design process the history or spectrum analysis for the same loading can be carried out economically a few times, for example, to study the stress history in different parts of the structure.

In the program, at completion of the eigensystem solution, all variables required for a response history or response spectrum analysis together with the frequencies and mode shapes are written on low speed storage. The program execution may be stopped at this stage and the information on low speed storage be copied to a physical tape. Later, this tape would be copied back to low speed storage before starting a response analysis. If, after a number of response analyses using the eigensystem on the tape, it is decided that more frequencies and mode shapes need be calculated, the information on the tape can be used to reduce the cost of the new eigensystem solution as described in Section 6.3.

7.5 Mode Superposition Versus Direct Integration

For an effective response history analysis the user must decide appropriately whether to use mode superposition or direct integration. It should be realized that the direct integration is equivalent to a mode superposition analysis in which all the eigenvalues and vectors have been calculated and the uncoupled equations in Eq. (17) with $p = n$ are integrated with a common time step Δt. Naturally, the integration can only be accurate for those modes for which Δt is smaller than a certain fraction of the period T. Using the Wilson θ-algorithm the integration errors effectively "filter" the high mode response, for which $\Delta t/T$ is large, out of the solution. This filtering is due to the amplitude decay observed in the numerical solution when $\Delta t/T$ is large. As an example, Fig. 8 shows the amplitude decay for the initial value problem indicated [6].

The effective filtering of the high frequency response from the solution may be beneficial. Integration accuracy cannot be obtained in the response of the modes for which $\Delta t/T$ is large and the filtering process allows one to obtain a total system solution in which the low mode response is accurately observed.

It is therefore noted that the direct integration is quite equivalent to a mode superposition analysis, in which only the lowest modes of the system, but a sufficient number to take proper account of the applied loading, are considered. The exact number of modes effectively included in the analysis depends on the time step size Δt and the distribution of the periods.

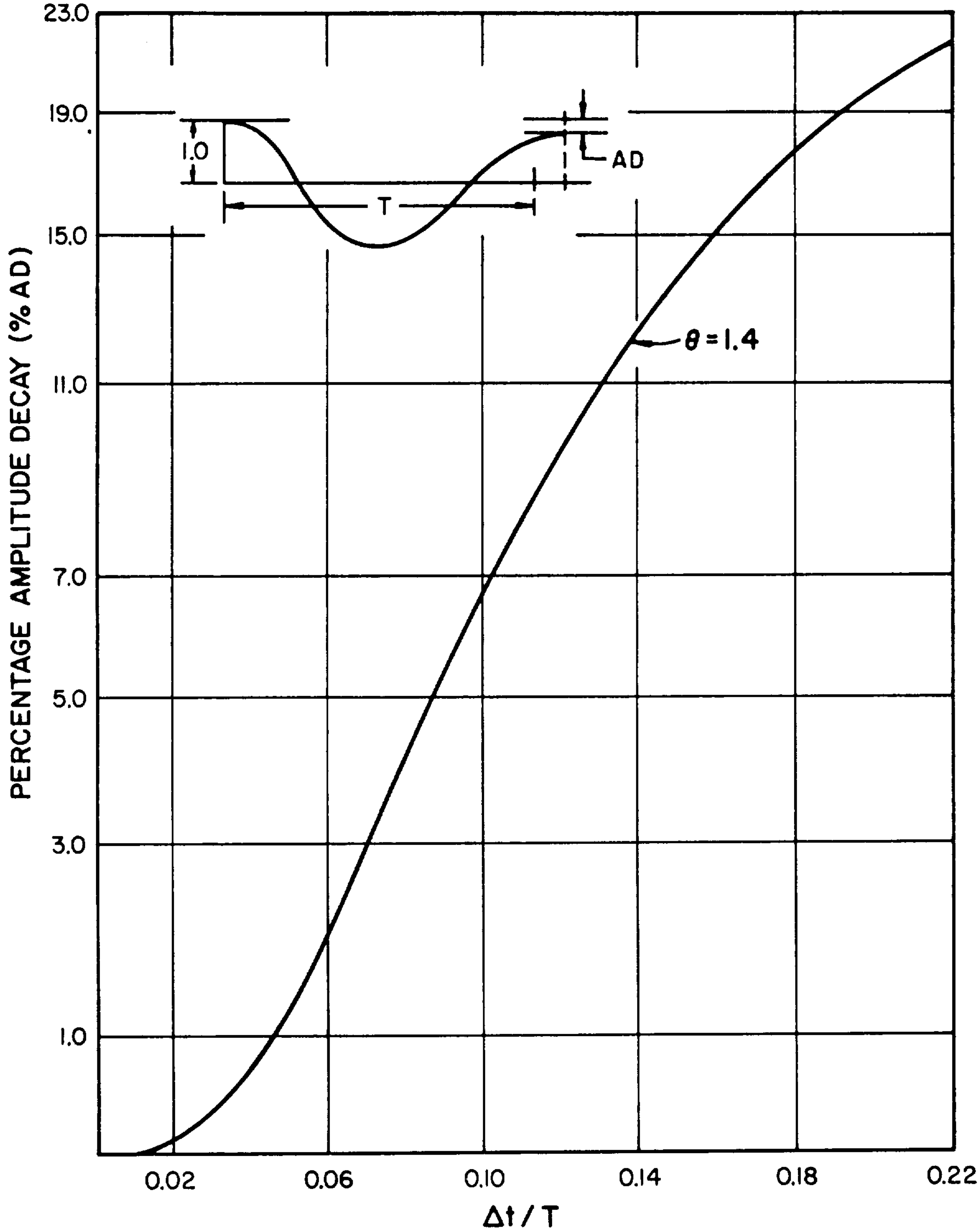


FIGURE 8: AMPLITUDE DECAY WILSON θ-METHOD

The advantages of mode superposition are essentially that frequencies and mode shapes are obtained and that a variety of response history and response spectrum analyses can be carried out with relatively small additional cost. Also, if the structure is slightly changed or more eigenvalues and vectors are required, i.e., the frequency domain to be considered shall be extended, the eigensystem solved for already can be used to reduce the cost of the new eigensystem solution (see Section 7.4).

The direct step-by-step integration, however, is more effective, when many modes need be included in the analysis and the response is required over relatively few time steps, such as in shock problems. It should be noted that the tape reading required in the direct integration analysis of large out-of-core systems can be costly because in the solution for the response in each time step the triangularized effective stiffness matrix must be taken into high speed storage.

8. DATA CHECK RUN

In the analysis of large structures it is important to be able to check the data read and generated by the program. For this purpose an option is given in which the program simply reads and generates all data, prints it and also writes the full data on low speed storage. At completion of data read and generation the information on low speed storage can be copied to a physical tape. This tape may then be used to plot the finite element mesh.

9. INSTALLATION OF SAP IV ON A SYSTEM OTHER THAN A CDC COMPUTER

SAP IV is written using FORTRAN IV and has been developed on a CDC computer. The program has also been installed with relatively little effort on IBM and UNIVAC machines.

The program or parts of it can essentially be used on any reasonably sized computer. SAP IV consists of about 14000 cards, and is organized in a standard Fortran overlay structure to reduce the required high speed storage for program execution.The main overlay essentially consists of the main program. The secondary overlays are, respectively, the element routines, the equation solver, the eigenvalue routines, the mode superposition history analysis program, the spectrum analysis program and the direct integration routine. Using only specific overlays efficient special purpose programs are obtained. For example, using the main overlay plus the secondary overlays of the pipe element, the eigenvalue routines and the response history analysis a special purpose pipe response history analysis program by mode superposition is obtained. On the CDC 6400 of the University of California, Berkeley, the complete program with 12000_{10} high speed storage locations allocated for solution processing, i.e. the blank common block A has a length of 12000, requires a field length of about 114000_8 for execution.

On installation of SAP IV on other machines than the CDC series, it must be observed that arithmetic calculations should be performed using about 14 digit words. This means that, for example, on IBM and UNIVAC machines double precision need be used. The calculations to be performed in double precision are in static and dynamic analysis the formation of element stiffness matrices, the formation of the structure stiffness matrix and main steps in the solution of the equations of motion, namely, the solution of $Ku = R$, the solution of the generalized eigenvalue problem $K\phi = \omega^2 M\phi$ and in the direct integration the solution of the effective displacements u_t^* (see Table 4). These calculations need primarily be performed in double precision because of truncation errors occurring when too few digits are used, which can cause large errors in the solution and numerical instabilities [20] [25].

With regard to the use of back-up storage, to keep the program system independent sequential accessing is used throughout. Therefore, since no advantage is taken of efficient buffering and direct access techniques, it need be noted that the use of secondary storage can be much improved when tailored to a specific system.

10. CONCLUDING REMARKS

The objective in this part of the report was to present a brief description of the computer program SAP IV. The program is a general analysis tool for the linear static and dynamic analysis of complex structures. While efficient in the solution process, however, it should be mentioned that pre- and post processing options have to a large extent not been developed; mainly, because the user is restricted to the particular peripheral equipment available to him.

With regard to the future of the program, various important improvements could be envisaged. The program does not have as yet substructure capabilities. More effective use of back-up storage could be achieved. The element routines could be further improved. A most important aspect are general error control procedures. In this area a significant amount of research is still required. Considering additional analysis capabilities, such as the use of consistent mass matrices, the possibility of including geometric and material nonlinearities, etc., it may be mentioned that a nonlinear static and dynamic analysis program is presently being developed [8].

- PART B -

SAMPLE ANALYSES

SAMPLE ANALYSES

In this part of the report brief problem descriptions for a set of standard data cases available with program SAP IV are given. Naturally, the few sample analyses can only demonstrate to a small degree the capabilities of the program. In general, detailed problem descriptions can be found in the references from which the sample analyses have been taken.

1. Static Analysis of Pipe Network

The pipe network shown in Fig. 9 corresponds to a sample problem solution presented in the User's Manual for the "ADLPIPE" piping analysis computer code [35]. The purpose of this analysis is to predict the static response of the system under the combined effects of:

(1) concentrated loads

(2) vertical (y-direction) gravity loads

(3) uniform temperature increase

(4) non-zero displacements imposed at one support point

Table 5 compares the reactions printed in the SAP and ADLPIPE solutions. The two solutions are in fair agreement; the SAP results satisfy equilibrium to all six digits, appearing in the printed output. In the table of applied loads, a total weight of 6284.03 lbs results from 950.686 inches of pipe weighing 6.61 lbs per inch.

2. Static Shell Analysis

The clamped spherical shell shown in Fig. 10 is analyzed for stresses produced by a uniform pressure applied on its outside surface. The SAP model represents a five degree wedge of the shell with eighteen

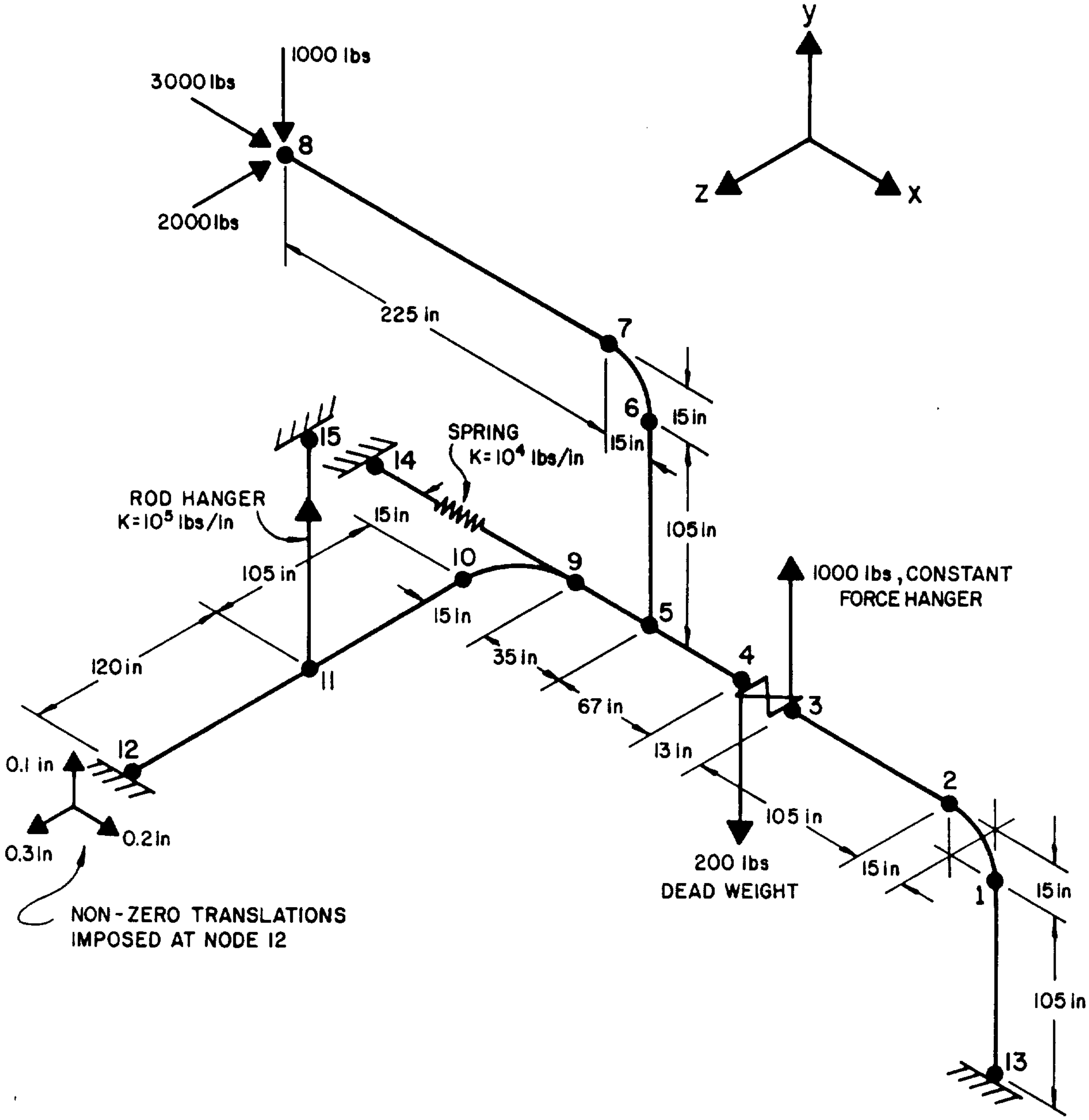


FIGURE 9: SAP MODEL OF PIPE NETWORK GIVEN IN ADLPIPE MANUAL

TABLE 5 FORCE EQUILIBRIUM SUMMARY
(SAP ANALYSIS OF ADLPIPE EXAMPLE 1)

A. REACTIONS

NODE	SAP			ADLPIPE		
	FX	FY	FZ	FX	FY	FZ
9	5643.51	.	.	5659.	.	.
11	.	-4044.59	.	.	-4052.	.
12	2350.08	4023.01	-4960.70	2361.	4026.	-4966.
13	-10993.59	4505.61	2960.70	-11021.	4509.	2966.
TOTAL	-3000.00	4484.03	-2000.00	-3001.	4483.	-2000.

B. APPLIED LOADS

LOADING TYPE	DIRECTION		
	X	Y	Z
CONCENTRATED:			
at node 3	.	1000.00	.
at node 4	.	-200.00	.
at node 8	3000.	1000.00	2000.
DISTRIBUTED WEIGHT:		-6284.03	
TOTAL	3000.	-4484.03	2000.

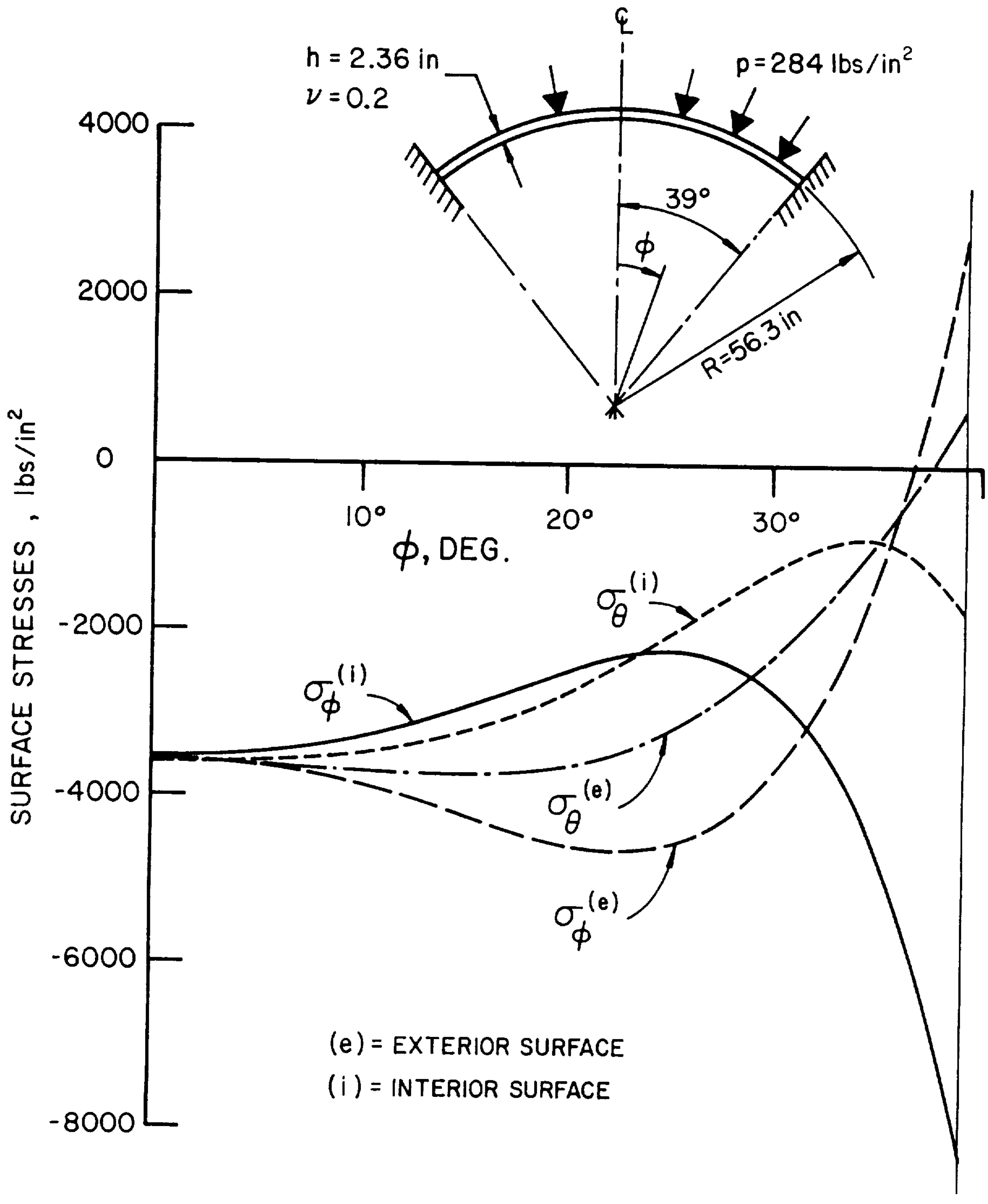


FIGURE 10: DISTRIBUTION OF SURFACE STRESSES IN A CLAMPED SPHERICAL SHELL UNDER EXTERNAL PRESSURE

thin shell elements along the thirty-nine degree meridian. The curves drawn in Fig. 10 are plots of meridian (ϕ) and circumferential (θ) direction surface stresses predicted by the SAP program at the element centroids.

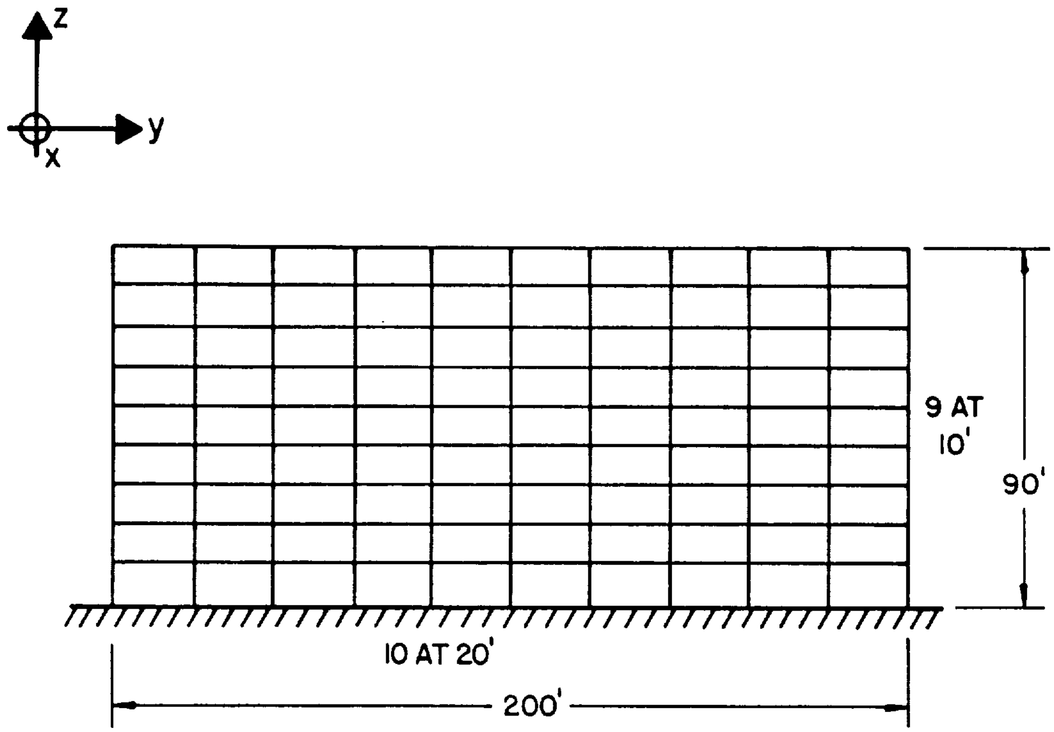
The solution of this problem is given in the text by Timoshenko [27], where the stress distribution of Fig. 10 may be found for comparison. It should be noted that program SAP calculates membrane stresses (force per unit area) and bending resultants (moment per unit length) from which the surface stresses in the figure have been evaluated.

3. Frequency and Mode Shape Analysis of Plane Frame

The lowest three frequencies and corresponding mode shapes of the plane frame shown in Fig. 11 are calculated. The results can be compared with the solutions published in references [4] [5]. Note that depending on the high speed storage available either a determinant search or a subspace iteration solution may be performed. The three lowest vibration periods of the frame are given in Table 6.

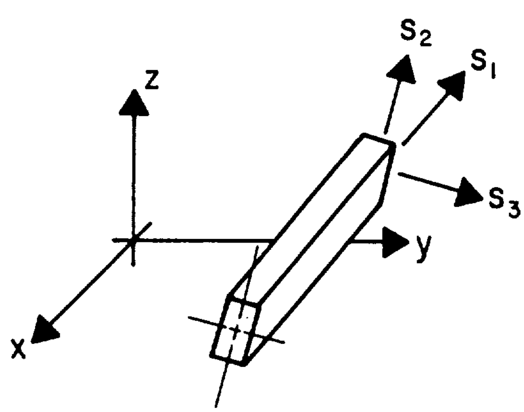
4. Response Spectrum Analysis of Pipe Network

A response spectrum analysis of the pipe assemblage shown in Fig. 12 is carried out. This is example 1 in the User's Manual for the "PIPDYN" computer program [36]. Good correspondence between the SAP and PIPDYN solutions is obtained. Table 7 compares local z-direction member end moments calculated by the two programs. In the analysis the lowest five modes are considered. Both, horizontal and vertical (proportional) spectra are simultaneously specified.



(a) ELEVATION OF FRAME

DATA : YOUNG'S MODULUS = 432000, MASS DENSITY = 1.0
FOR ALL BEAMS AND COLUMNS $A_1 = 3.0$, $I_1 = I_2 = I_3 = 1.0$
UNITS : FT, KIPS



(b) BEAM ELEMENT DEFINITION

S_1, S_2 AND S_3 = BEAM LOCAL AXES

I_1, I_2 AND I_3 = FLEXURAL INERTIA ABOUT S_1, S_2, AND S_3

A_1 = AREA ASSOCIATED WITH S_1

FIGURE 11: SAP MODEL OF PLANE FRAME

TABLE 6 PERIODS OF PLANE FRAME

MODE NUMBER	PERIOD (SEC)
1	8.183
2	2.673
3	1.543

TABLE 7 COMPARISON OF MOMENT PREDICTIONS
(SAP ANALYSIS OF PIPDYN EXAMPLE 1)

ELEMENT NUMBER	MOMENT MZ (Kip in) IN ELEMENT LOCAL COORDINATES (at element ends 1, see Ref. 29 pp. 54)	
	SAP	PIPDYN
1	376.9	377.0
2	30.67	30.68
3	152.9	152.9
4	100.6	100.6
5	83.27	83.27
6	46.17	46.19
7	1.081	1.082
8	21.59	21.81
9	7.052	7.038
10	7.537	7.571
11	160.3	160.4
12	78.07	78.09
13	26.08	25.80

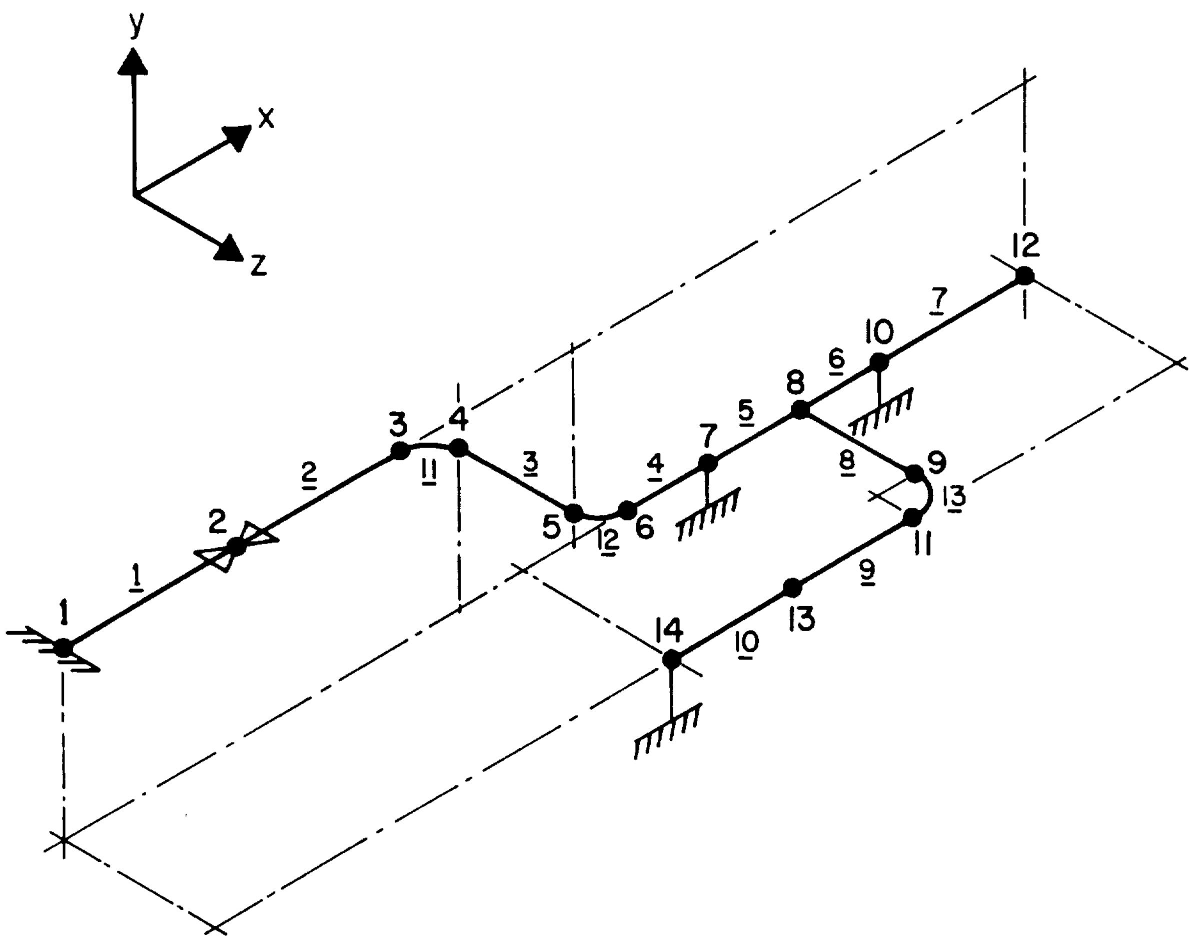


FIGURE 12: SAP MODEL OF PIPDYN EXAMPLE 1,
RESPONSE SPECTRUM ANALYSIS

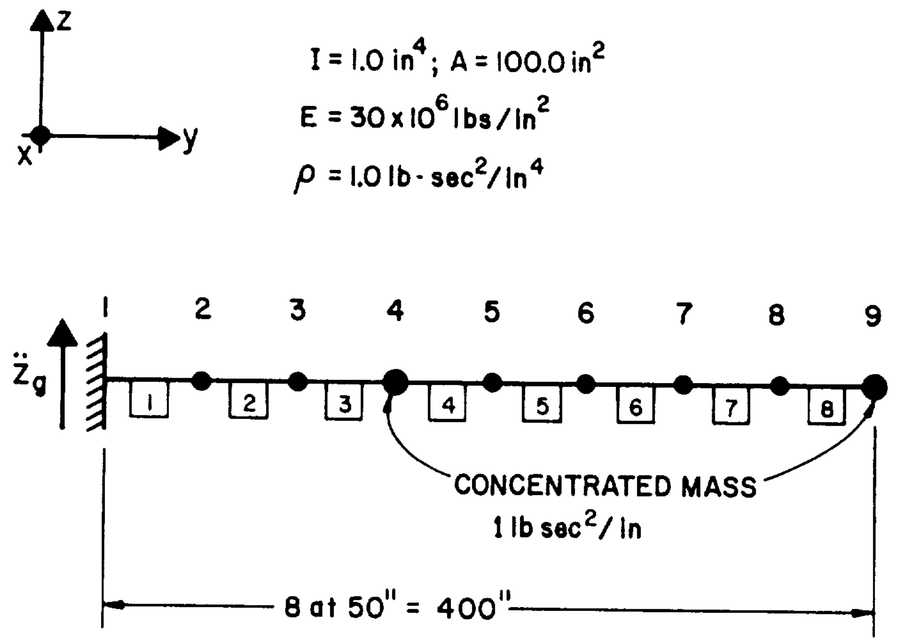
5. Mode Superposition Time History Response Analysis of Cantilever

The cantilever beam shown in Fig. 13 is analyzed for the ground acceleration shown in the same figure. The solution to this problem is obtained independently using the "DRA2" computer code [21]. This program calculates the dynamic response by direct integration of the (coupled) equations of motion using the Wilson θ-algorithm [6].

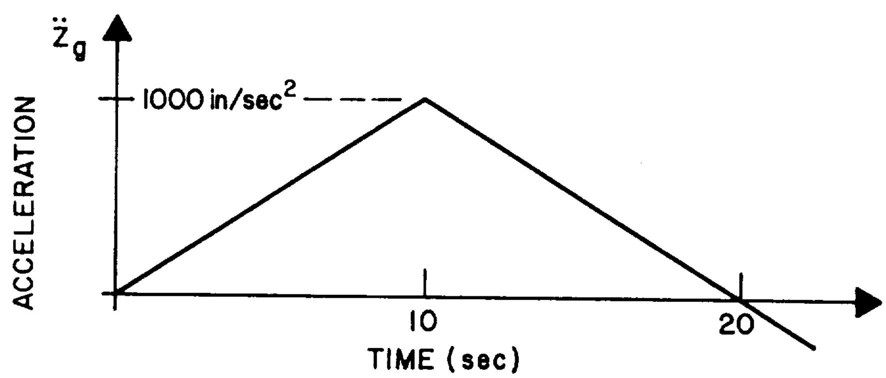
The response history of the beam model is evaluated in SAP using mode superposition including all eight flexural modes developed in the cantilever; Table 8 lists the periods of these eight modes computed by SAP. Figure 14 shows the variation of the transverse displacements and of the fixed-end moment calculated by SAP. The DRA2 predictions agree with the SAP results to 5 or more digits and, consequently, are not shown for comparison.

6. Mode Superposition Time History Response Analysis of Cylindrical Tube

The response of the simply supported cylindrical tube shown in Fig. 15 for a suddenly applied load is calculated by mode superposition. Using symmetry one half of the tube is idealized as an assemblage of axisymmetric elements with a total of 61 degrees of freedom. In the mode superposition analysis only the lowest twenty modes are considered; some of the vibration periods are listed in Table 9. Figure 15 shows a comparison of the radial displacements calculated by the program with a Timoshenko-Love solution [24].



(a) NODE AND BEAM NUMBER ASSIGNMENTS FOR THE CANTILEVER MODEL



(b) GROUND ACCELERATION APPLIED AT NODE 1

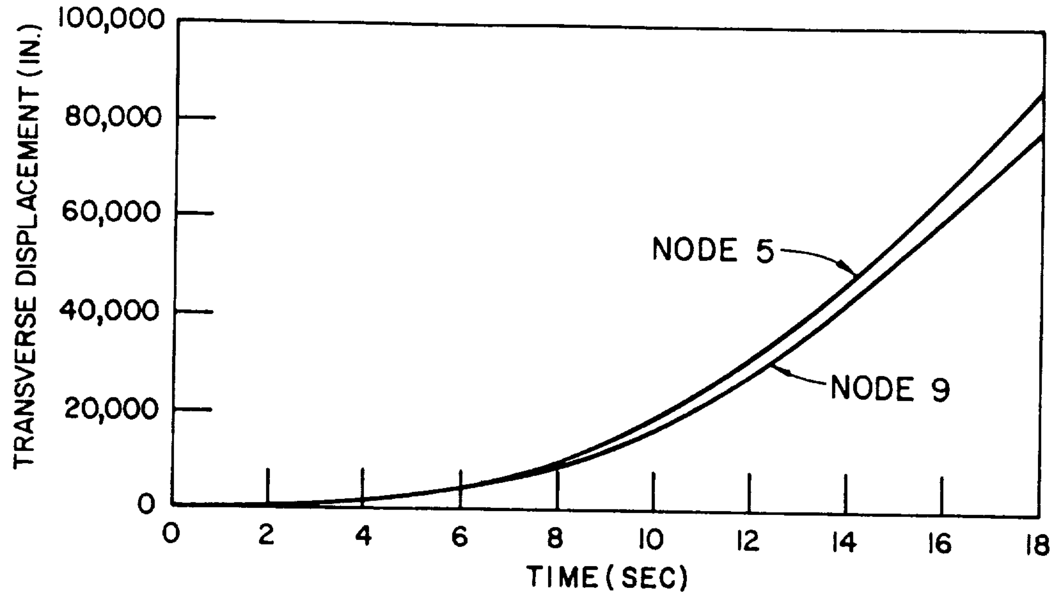
FIGURE 13: RESPONSE HISTORY ANALYSIS OF CANTILEVER BEAM

TABLE 8 CANTILEVER BEAM ANALYSIS - NATURAL PERIODS FOR THE EIGHT (LOWEST) FLEXURAL MODES

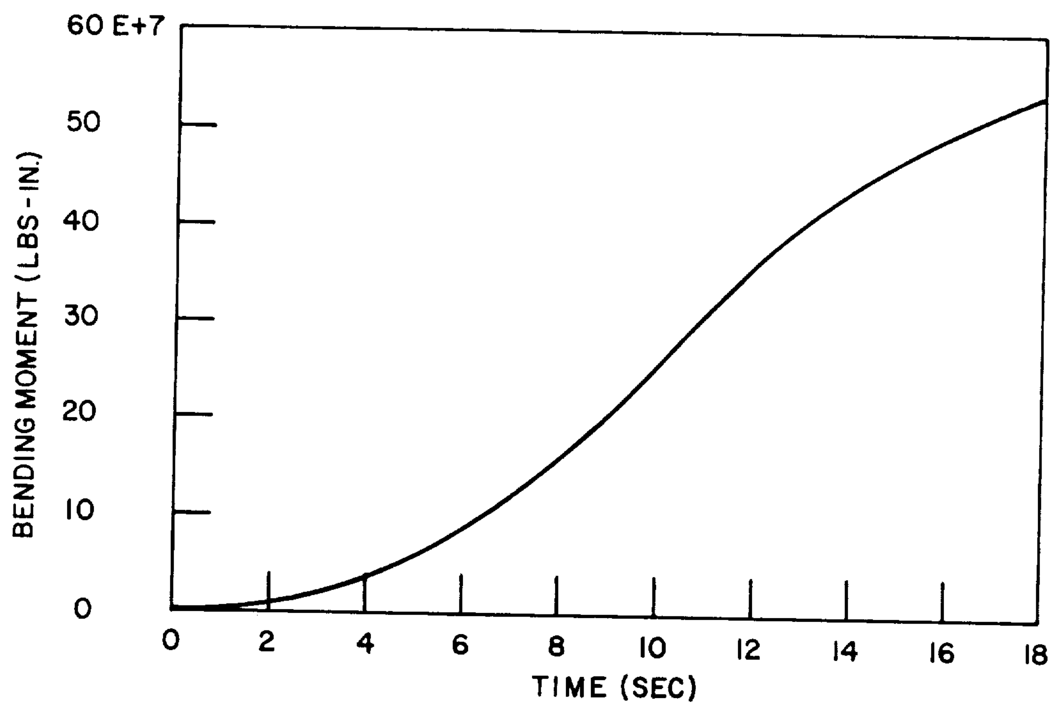
MODE NUMBER	PERIOD (SEC)
1	525.79
2	85.368
3	30.965
4	16.059
5	9.9006
6	6.8276
7	5.1865
8	4.3777

TABLE 9 CYLINDRICAL TUBE ANALYSIS - SOME NATURAL PERIODS

MODE NUMBER	PERIOD (SEC x 10^{-3})
1	1.2788
5	0.62140
10	0.32983
15	0.17463
20	0.11497

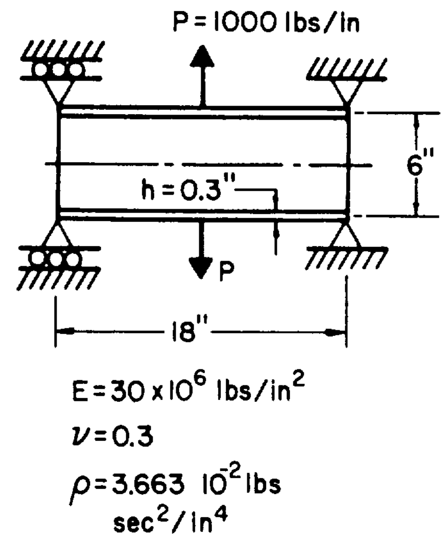


(a) TRANSVERSE DEFLECTIONS

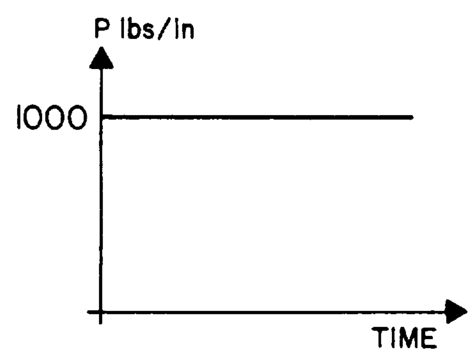


(b) MOMENT AT NODE 1
(FIXED END OF CANTILEVER)

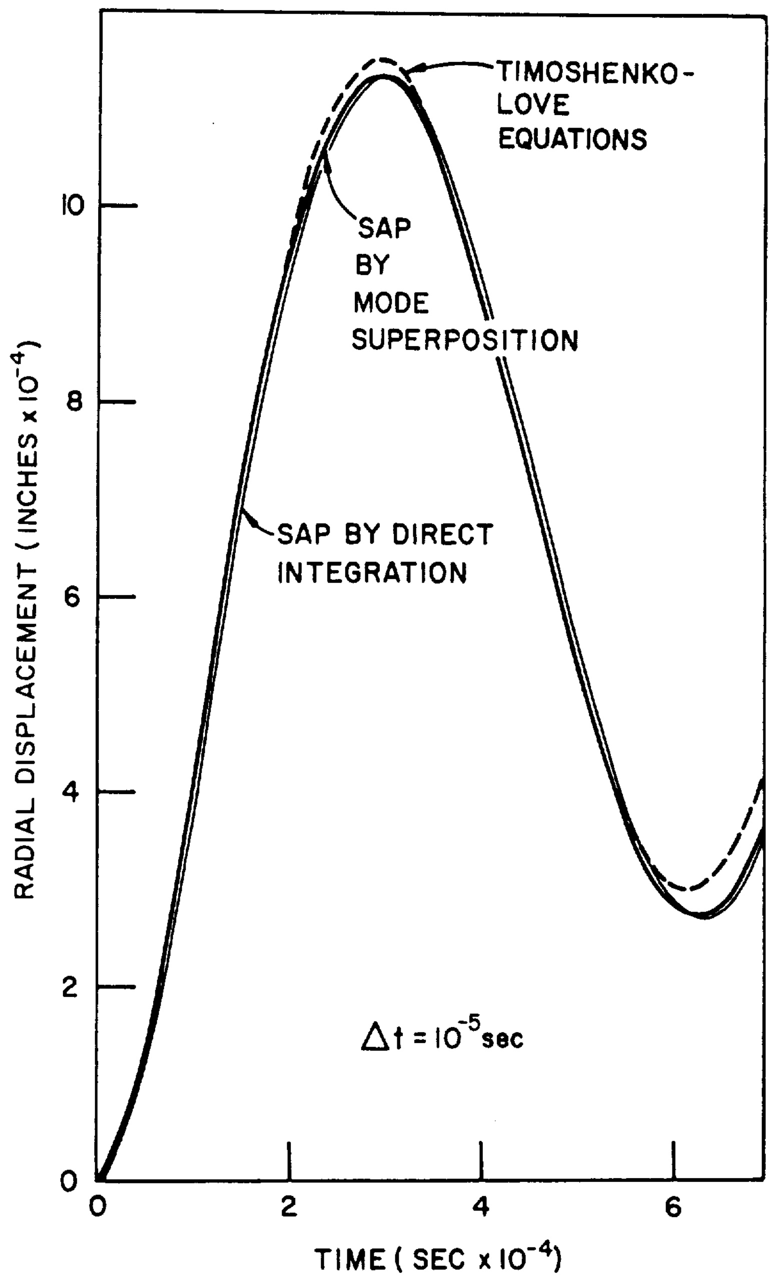
FIGURE 14: CANTILEVER RESPONSE



a) CYLINDRICAL TUBE



b) TIME VARIATION OF LOAD



c) RADIAL DISPLACEMENT VERSUS TIME

FIGURE 15: RESPONSE HISTORY ANALYSIS OF CYLINDRICAL TUBE

7. Direct Integration Time History Response Analysis of Cylindrical Tube

The response of the simply supported tube shown in Fig. 15 for the applied load is calculated by direct integration. The same finite element idealization and time step Δt as in the mode superposition is used. Figure 15 shows the radial displacements as calculated by the program.

REFERENCES

1. Argyris, J. H., and Kelsey, A., "Energy Theorems and Structural Analysis," Aircraft Engineering, Vol. 31, Oct. and Nov. 1954, Feb. to May 1955. Also published by Butterworth's Scientific Publications, London, 1960.

2. Argyris, J. H., "Continua and Discontinua," Proceedings Conference on Matrix Methods in Structural Mechanics, Wright Patterson AFB, Ohio, 1965.

3. Bathe, K. J., and Wilson, E. L., "Solution Methods for Eigenvalue Problems in Structural Mechanics," Int. J. Num. Methods in Engg., Vol. 6, No. 2, 1973.

4. Bathe, K. J., and Wilson, E. L., "Eigensolution of Large Structural Systems with Small Bandwidth," ASCE Journal of Eng. Mech. Div., June, 1973.

5. Bathe, K. J., and Wilson, E. L., "Large Eigenvalue Problems in Dynamic Analysis," ASCE Journal of Eng. Mech. Div., Dec. 1972.

6. Bathe, K. J., and Wilson, E. L., "Stability and Accuracy Analysis of Direct Integration Methods," Int. J. of Earthquake Engg. and Struct. Dynamics, Vol. 1, No. 2, 1973.

7. Bathe, K.J., and Wilson, E.L., "Thick Shell Structures", Proceedings International Symposium on Structural Mechanics Software, University of Maryland, College Park, Maryland, June 1974.

8. Bathe, K.J., Wilson, E.L., and Iding, R.H., "NONSAP - A Structural Analysis Program for Static and Dynamic Response of Nonlinear Systems", SESM Report 74-3, Department of Civil Engineering, University of California, Berkeley, 1974.

9. Clough, R. W., "Analysis of Structural Vibrations and Dynamic Response", Proceedings 1st U.S.-Japan Symposium on Recent Advances in Matrix Methods of Structural Analysis and Design, Tokyo, Japan, 1968.

10. Clough, R. W., "Earthquake Analysis by Response Spectrum Superposition," Bulletin of the Seismological Society of America, Vol. 52, July 1962.

11. Clough, R. W., and Bathe, K. J., "Finite Element Analysis of Dynamic Response," Proceedings 2nd US-Japan Symposium on Recent Advances in Computational Methods of Structural Analysis and Design, Berkeley, California, 1972.

12. Clough, R. W., and Felippa, C. A., "A Refined Quadrilateral Element for Analysis of Plate Bending," Proceedings 2nd Conference on Matrix Methods in Structural Mechanics, Wright Patterson AFB, Ohio, 1968.

13. Clough, R. W., and Wilson, E. L., "Dynamic Finite Element Analysis of Arbitrary Thin Shells," Computers and Structures, Vol. 1, No.1, 1971.

14. Felippa, C. A., "Refined Finite Element Analysis of Linear and Nonlinear Two-dimensional Structures," SESM Report 66-2, Dept. of Civil Engineering, University of California, Berkeley, 1966.

15. Felippa, C. A., and Clough, R. W., "The Finite Element Method in Solid Mechanics," Proceedings Symposium on Numerical Solutions of Field Problems in Continuum Mechanics, Durham, North Carolina, 1968.

16. Hall, A. S., Tezcan, S. S., and Bulent, D., Discussion of paper "Curved Beam Stiffness Coefficients," ASCE Journal of Struct. Div., Feb., 1969.

17. Hurty, W., and Rubinstein, M. F., <u>Dynamics of Structures</u>, Prentice Hall, Inc., 1964

18. Irons, B. M., "Structural Eigenvalue Problems: Elimination of Unwanted Variables," Journal A.I.A.A., Vol. 3, 1965.

19. Irons, B. M., "Numerical Integration Applied to Finite Element Methods," Conf. on Use of Digital Computers in Structural Engineering, University of New Castle, England, July 1966.

20. MacNeal, R.H., "The NASTRAN Theoretical Manual", NASA Report No. NASA SP-221, September 1970.

21. Peterson, F. E., and Bathe, K. J., "Nonlinear Dynamic Analysis of Reactor Core Components," Report S-104.3, Engineering/Analysis Corporation, Berkeley, California, March 1972.

22. Poley, S., "Mesh Analysis of Piping Systems," IBM New York Scientific Center Technical Report No. 320-2939, March 1968.

23. Przemieniecki, J. S., <u>Theory of Matrix Structural Analysis</u>, McGraw-Hill, New York, 1968.

24. Reismann, H., and Padlog, J., "Forced, Axisymmetric Motions of Cylindrical Shells," Journal of the Franklin Institute, Vol. 284, No. 5, Nov. 1967.

25. Roy, J. R., "Numerical Errors in Structural Solutions," ASCE Journal of the Structural Division, April 1971.

26. Strang, G., and Fix, G.J., "An Analysis of the Finite Element Method", Prentice Hall, Inc., Englewood Cliffs, New Jersey, 1973.

27. Timoshenko, S., Theory of Plates and Shells, 2nd Edition, McGraw-Hill, 1959, pp. 544.

28. Wilson, E. L., "SAP - A General Structural Analysis Program," SESM Report 70-20, Dept. of Civil Engineering, University of California, Berkeley, 1970.

29. Wilson, E. L., "SOLID SAP - A Static Analysis Program for Three-Dimensional Solid Structures," SESM Report 71-19, Dept. of Civil Engineering, University of California, Berkeley, 1971.

30. Wilson, E. L., "Earthquake Analysis of Reactor Structures," Proceedings Symposium on Seismic Analysis of Pressure Vessels and Piping Components, The American Society of Mechanical Engineers, 1971.

31. Wilson, E. L., Bathe, K. J., and Doherty, W. P., "Direct Solution of Large Systems of Linear Equations," Computers and Structures, to appear.

32. Wilson, E. L., Taylor, R. L., Doherty, W. P., and Ghaboussi, J., "Incompatible Displacement Models," ONR Symposium on Matrix Methods in Structural Mechanics, University of Illinois, Urbana, Illinois, Sept. 1971.

33. Wilson, E. L., and Penzien, J., "Evaluation of Orthogonal Damping Matrices," Int. J. for Num. Methods in Engg., Vol. 4, No. 1, 1972.

34. Zienkiewicz, O. C., The Finite Element Method in Engineering Science, McGraw-Hill, 1971.

Computer Program Manuals:

35. "ADL Pipe Static-Thermal-Dynamic Pipe Stress Analysis," Arthur D. Little, Inc., Cambridge, Massachusetts, January 1971.

36. "Construction Industry Programs, PIPDYN: Dynamic Analysis of Piping Systems," Computer Sciences Corporation, Los Angeles, California.

- PART C -

APPENDICES

APPENDIX - DATA INPUT TO SAP IV

I. HEADING CARD (12A6)

notes	columns	variable	entry
(1)	1 - 72	HED(12)	Enter the heading information to be printed with the output

NOTES/

(1) Begin each new data case with a new heading card.

II. MASTER CONTROL CARD (8I5)

notes	columns	variable	entry
(1)	1 - 5	NUMNP	Total number of nodal points (joints) in the model
(2)	6 - 10	NELTYP	Number of element groups
(3)	11 - 15	LL	Number of structure load cases; GE.1; static analysis EQ.0; dynamic analysis
(4)	16 - 20	NF	Number of frequencies to be found in the eigenvalue solution; EQ.0; static analysis, No freq. constraint GE.1; dynamic analysis, or [illegible] constraint
(5)	21 - 25	NDYN	Analysis type code: EQ.0; static analysis EQ.1; eigenvalue/vector solution EQ.2; forced dynamic response by mode superposition EQ.3; response spectrum analysis EQ.4; direct step-by-step integration EQ.5 [illegible]
(6)	26 - 30	MODEX	Program execution mode: EQ.0; problem solution EQ.1; data check only
(7)	31 - 35	NAD	Total number of vectors to be used in a SUBSPACE INTERATION solution for eigenvalues/vectors: EQ.0; default set to: MIN{2*NF,NF+8}
(8)	36 - 40	KEQB	Number of degrees of freedom (equations) per block of storage: EQ.0; calculated automatically by the program
(9)	41 - 45	NIOSV	
(10)	46 - 50	[illegible]	= Number of design iterations
(11)	51 - 55	NDAM	

NOTES/

(1) Nodes are labeled with integers ranging from "1" to the total number of nodes in the system, "NUMNP". The program exits with no diagnostic message if NUMNP is zero (0). Thus, two blank cards are used to end the last data case in a run; i.e., one blank heading card (Section I) and one blank card for this section.

(2) For each different element type (TRUSS, BEAM, etc.) a new element group need be defined. Elements within groups are assigned integer labels ranging from "1" to the total number of elements in the group. Element groups are input in Section IV, below.

Element numbering must begin with one (1) in each different group. It is possible to use more than one group for an element type. For example, all columns (vertical beams) of a building may be considered one group and the girders (horizontal beams) may be considered another group.

(3) At least one (1) load condition must be specified for a static (NDYN.EQ.0) analysis. If the data case calls for one of the dynamic analysis options (NDYN.EQ.1, 2, 3, or 4), no load cases can be requested (i.e., LL is input as "0"). The program always processes Sections V (Concentrated Load/Mass Data) and VI (Element Load Multipliers) and expects to read some data. For the case of a dynamic analysis (NDYN.GE.1) only mass coefficients can be input in Section V, and one (1) blank element load multiplier card is expected in Section VI.

(4) For a static analysis, NF.EQ.0. If NDYN.EQ.1, 2 or 3, the lowest NF eigenvalues are determined by the program. Note that a dynamic solution may be re-started after eigenvalue extraction (providing a previous eigenvalue solution for the model was saved on tape as described in Appendix A). NF for the original and re-start runs must be the same.

(5) If NDYN.EQ.2 or NDYN.EQ.3 the program first solves for NF eigenvalues/vectors and then performs the forced response solution (or the response spectrum analysis). Thus, the program expects to read the control card governing the eigensolution (Section VII.A) before reading data in either Sections VII.B or VII.C. For the case NDYN.EQ.1, the program solves for NF eigenvalues/vectors, prints the results and proceeds to the next data case. The results for the eigenvalue solution phase (NDYN.EQ.1) may be saved for later use in automatic re-start (Appendix A lists the control cards that are required to affect this save operation), i.e. a dynamic solution may be restarted without repeating the solution for modes and frequencies. If this data case is a re-start job, set NDYN.EQ.-2 for a forced response solution, or set NDYN.EQ.-3 for a response spectrum analysis. Note that the solution may be re-started a multiple of times (to run different ground spectra or different time-dependent forcing functions) because the program does not destroy the contents of the re-start tape.

If NDYN.EQ.4 the program performs the response solution by direct step-by-step integration and no eigenvalue solution control card should be provided.

II. **MASTER CONTROL CARD** (continued)

(6) In the data-check-only mode (MODEX.EQ.1), the program writes only one file, "TAPE8", and this file may be saved for use as input to special purpose programs such as mesh plotters, etc. TAPE8 contains all data input in its completely generated form. If MODEX.EQ.1, most of the expensive calculations required during normal (MODEX.EQ.0) execution are passed. TAPE8, however, is not written during normal problem solution.

Note that a negative value for NDYN ("-2" or "-3"), when executing in the data-check-only mode, does not cause the program to read the re-start tape which contains the eigensolution information; instead, the program jumps directly from this card to Section VII.B (or Section VII.C) and continues reading and checking data cards without performing the solution.

(7) If the program is to solve for eigenvalues using the SUBSPACE ITERATION algorithm, the entry in cc 31-35 can be used to change the total number of iteration vectors to be used from the default minimum of 2*NF or NF+8 (whichever is smaller) to the value "NAD". The effect of increasing NAD over the default value is to accelerate convergence in the calculations for the lowest NF eigenvalues. NAD is principally a program testing parameter and should normally be left blank.

(8) KEQB is a program testing parameter which allows the user to test multiple equation block solutions using small data cases which would otherwise be one block problems. KEQB is normally left blank.

III. NODAL POINT DATA (A1,I4,6I5,3F10.0,I5,F10.0)

notes	columns	variable	entry
(1)	1	CT	Symbol describing coordinate system for this node; EQ. ; (blank) cartesian (X,Y,Z) EQ.C; cylindrical (R,Y,θ)
(2)	2 - 5	N	Node number
(3)	6 - 10	IX(N,1)	X-translation boundary condition code
	11 - 15	IX(N,2)	Y-translation boundary condition code
	16 - 20	IX(N,3)	Z-translation boundary condition code
	21 - 25	IX(N,4)	X-rotation boundary condition code
	26 - 30	IX(N,5)	Y-rotation boundary condition code
	31 - 35	IX(N,6)	Z-rotation boundary condition code EQ.0; free (loads allowed) EQ.1; fixed (no load allowed) GT.1; master node number (beam nodes only)
(4)	36 - 45	X(N)	X (or R) -ordinate
	46 - 55	Y(N)	Y -ordinate
	56 - 65	Z(N)	Z (or θ) -ordinate (degrees)
(5)	66 - 70	KN	Node number increment
(6)	71 - 80	T(N)	Nodal temperature

NOTES:

(1) A special cylindrical coordinate system is allowed for the global description of nodal point locations. If a "C" is entered in card column one (1), then the entries given in cc 36-65 are taken to be references to a global (R,Y,θ) system rather than to the standard (X,Y,Z) system. The program converts cylindrical coordinate references to cartesian coordinates using the formulae:

$$X = R \sin\theta$$
$$Y = Y$$
$$Z = R \cos\theta$$

Cylindrical coordinate input is merely a user convenience for locating nodes in the standard (X,Y,Z) system, and no other references to the cylindrical system are implied; i.e., boundary condition specifications, output displacement components, etc. are referenced to the (X,Y,Z) system.

(2) Nodal point data must be defined for all (NUMNP) nodes. Node data may be input directly (i.e., each node on its own individual card) or the generation option may be used if applicable (see note 5, below).

Admissible nodal point numbers range from "1" to the total number of nodes "NUMNP". Illegal references are: N.LE.0 or N.GT.NUMNP.

(3) Boundary condition codes can only be assigned the following values (M = 1,2,...,6):

IX(N,M) = 0;	unspecified (free) displacement (or rotation) component
IX(N,M) = 1;	deleted (fixed) displacement (or rotation) component
IX(N,M) = K;	node number "K" (1 < K ≥ NUMNP and K ≠ N) is the "master" node to which the Mth degree of freedom at node "N" is a "slave"

An unspecified (IX(N,M) = 0) degree of freedom is free to translate or rotate as the solution dictates. Concentrated forces (or moments) may be applied (Section V, below) in this degree of freedom. One (1) system equilibrium equation is required for each unspecified degree of freedom in the model. The maximum number of equilibrium equations is always less than six (6) times the total number of nodes in the model.

Deleted (IX(N,M) = 1) degrees of freedom are removed from the final set of equilibrium equations. Deleted degrees of freedom are fixed (points of reaction), and any loads applied in these degrees of freedom are ignored by the program. Nodes that are used for geometric reference only (i.e., nodes not assigned to any element) must have all six (6) degrees of freedom deleted. Nodal degrees of freedom having undefined stiffness (such as rotations in an all TRUSS model, out-of-plane components in a two-dimensional planar model, etc.) should be deleted. Deletions have the beneficial effect of reducing the size of the set of equations that must be solved. The table below lists the types of degrees of freedom that are defined by each different element type. The table was prepared assuming that the element has general orientation in (X,Y,Z) space.

ELEMENT TYPE	DEGREES OF FREEDOM WITH DEFINED STIFFNESS					
	δX	δY	δZ	$\delta\theta_X$	$\delta\theta_Y$	$\delta\theta_Z$
1. TRUSS	x	x	x			
2. BEAM	x	x	x	x	x	x
3. MEMBRANE	x	x	x			
4 2D QUADRILATERAL		x	x			
5. 3D BRICK	x	x	x			
6. PLATE SHELL	x	x	x	x	x	x
7. BOUNDARY	x	x	x	x	x	x

III. NODAL POINT DATA (continued)

ELEMENT TYPE	δX	δY	δZ	$\delta\theta_X$	$\delta\theta_Y$	$\delta\theta_Z$
	DEGREES OF FREEDOM WITH DEFINED STIFFNESS					
8. THICK SHELL	x	x	x			
9. 3D/PIPE	x	x	x	x	x	x

Hence, for an all 3D/BRICK model, only the X,Y,Z translations are defined at the node, and the number of equations can be cut in half by deleting the three (3) rotational components at every node. If a node is common to two or more different element types, then the non-trivial degrees of freedom are found by combination. For example, all six (6) components are possible at a node common to both BEAM and TRUSS elements; i.e., the BEAM governs.

A "master/slave" option is allowed to model rigid links in the system. For this case, IX(N,M) = K means that the Mth degree of freedom at node "N" is "slave" to (dependent on) the same (Mth) degree of freedom at node "K"; node "K" is said to be the master node to which node N is slave. Note that no actual beam need to run from node K to node N, however the following restrictions hold:

(a) Node one (1) cannot be a master node; i.e., $K \neq 1$.
(b) Nodes "N" and "K" must be beam-only nodes; i.e., no other element type may be connected to either node N or K.
(c) A node "N" can be slave to only one master node, "K"; multiple nodes, however, can be slave to the same master.
(d) If the beam from "N" to "K" is to be a rigid link arbitrarily oriented in the X,Y,Z space, then all six (6) degrees of freedom at node "N" must be made slaves to node "K"

Displacement/rotation components for slave degrees of freedom at node "N" are not recovered for printing; i.e., zeroes appear as output for slave degrees of freedom.

(4) When CT (Col. 1) is equal to the character "C", the values input in CC 36-65 are interpreted as the cylindrical (R,Y,θ) coordinates of node "N". Y is the axis of symmetry. R is the distance of a point from the Y-axis. The angle θ is measured clockwise from the positive Z-axis when looking in the positive Y direction. The cylindrical coordinate values are printed as entered on the card, but immediately after printing the

global cartesian values are computed from the input entries. Note that boundary condition codes always refer to the the (X,Y,Z) system even if the node happens to be located with cylindrical coordinates.

(5) Nodal point cards need not be input in node-order sequence; eventually, however, all nodes in the integer set {1, NUMNP} must be defined. Joint data for a series of nodes

$$\{N_1,\ N_1+1 \times KN_2,\ N_1+2 \times KN_2, \ldots, N_2\}$$

may be generated from information given on two (2) cards in sequence:

CARD 1 / N_1, IX(N_1,1),..., IX(N_1,6), X(N_1),..., KN_1, T(N_1)/

CARD 2 / N_2, IX(N_2,1),..., IX(N_2,6), X(N_2),..., KN_2, T(N_2)/

KN_2 is the mesh generation parameter given on the second card of a sequence. The first generated node is $N_1+1 \times KN_2$; the second generated node is $N_1+2 \times KN_2$, etc. Generation continues until node number $N_2 - KN_2$ is established. Note that the node difference $N_2 - N_1$ must be evenly divisible by KN_2. Intermediate nodes between N_1 and N_2 are located at equal intervals along the straight line between the two points. Boundary condition codes for the generated data are set equal to the values given on the first card. Node temperatures are found by linear interpolation between T(N_1) and T(N_2). Coordinate generation is always performed in the (X,Y,Z) system, and no generation is performed if KN_2 is zero (blank).

(6) Nodal temperatures describe the actual (physical) temperature distribution in the structure. Average element temperatures established from the nodal values are used to select material properties and to compute thermal strains in the model (static analysis only).

IV. ELEMENT DATA

TYPE 1 - THREE-DIMENSIONAL TRUSS ELEMENTS

Truss elements are identified by the number 1. Axial forces and stresses are calculated for each member. A uniform temperature change and inertia loads in three directions can be considered as the basic element load conditions. The truss elements are described by the following sequence of cards:

A. Control Card (3I5)

Columns		
	1 - 5	The number 1
	6 - 10	Total number of truss elements
	11 - 15	Number of material property cards
	16 - 20	TYPE OF CROSS SECTION

B. Material Property Cards (I5,5F10.0)

There need be as many of the following cards as are necessary to define the properties listed below for each element in the structure.

Columns		
	1 - 5	Material identification number
	6 - 15	Modulus of elasticity
	16 - 25	Coefficient of thermal expansion
	26 - 35	Mass density (used to calculate mass matrix)
	36 - 45	Cross-sectional area
	46 - 55	Weight density (used to calculate gravity loads)

C. Element Load Factors (4F10.0) Four cards

Three cards specifying the fraction of gravity (in each of the three global coordinate directions) to be added to each element load case.

Card 1: Multiplier of gravity load in the +X direction

Columns		
	1 - 10	Element load case A
	11 - 20	Element load case B
	21 - 30	Element load case C
	31 - 40	Element load case D

Card 2: As above for gravity in the +Y direction

Card 3: As above for gravity in the +Z direction

Card 4: This indicates the fraction of the thermal load to be added to each of the element load cases.

D. Element Data Cards (4I5,F10.0,I5)

One card per element in increasing numerical order starting with one.

Columns 1 - 5 Element number

Columns	6 - 10	Node number I
	11 - 15	Node number J
	16 - 20	Material property number
	21 - 30	Reference temperature for zero stress
	31 - 35	Optional parameter k used for automatic generation of element data.
	36 - 45	AREA
	46 - 50	IGROUP

NOTES/

(1) If a series of elements exist such that the element number, N_i, is one greater than the previous element number (i.e. $N_i = N_{i-1} + 1$) and the nodal point number can be given by

$$I_i = I_{i-1} + k$$

$$J_i = J_{i-1} + k$$

then only the first element in the series need be provided. The element identification number and the temperature for the generated elements are set equal to the values on the first card. If k (given on the first card) is input as zero it is set to 1 by the program.

(2) The element temperature increase ΔT used to calculate thermal loads is given by

$$\Delta T = (T_i + T_j)/2.0 - T_r$$

where $(T_i + T_j)/2.0$ is the average of the nodal temperatures specified on the nodal point data cards for nodes i and j; and T_r is the zero stress reference temperature specified on the element card. For truss elements it is generally more convenient to set $T_i = T_j = 0.0$ such that $\Delta T = -T_r$ (note the minus sign). Other types of member loadings can be specified using an equivalent ΔT. If a truss member has an initial lack of fit by an amount d (positive if too long) then $\Delta T = d/(\alpha L)$. If an initial prestress force P (positive if tensile) is applied to the member ends that is released after the member is connected to the rest of the structure then $\Delta T = - P/(\alpha A E)$. In the above formulas A = cross section area, L = member length and α = coefficient of thermal expansion.

IV. ELEMENT DATA (continued)

TYPE 2 - THREE-DIMENSIONAL BEAM ELEMENTS

Beam elements are identified by the number 2. Forces (axial and shear) and moments (bending and torsion) are calculated (in the beam local coordinate system) for each beam. Gravity loadings in each coordinate direction and specified fixed end forces form the basic element load conditions.

The beam elements are described by the following sequence of cards:

A. Control Card (5I5)

Columns	1 - 5	The number 2
	6 - 10	Total number of beam elements
	11 - 15	Number of element property cards
	16 - 20	Number of fixed end force sets
	21 - 25	Number of material property cards

B. Material Property Cards (I5,3F10.0)

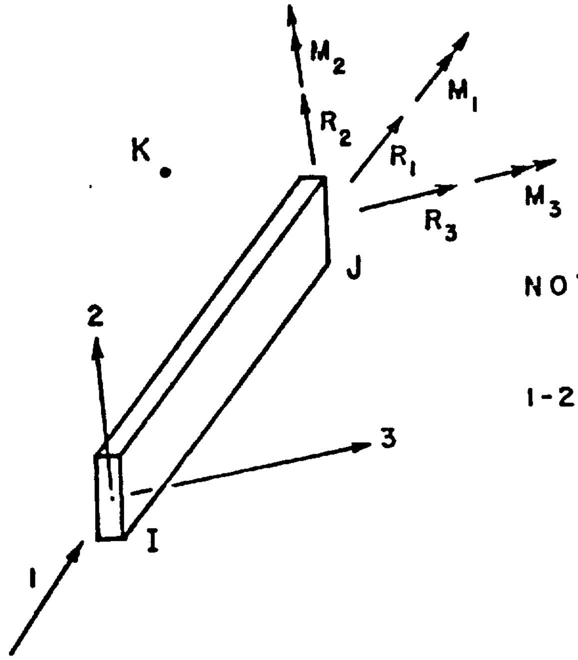
Columns	1 - 5	Material identification number
	6 - 15	Young's modulus
	16 - 25	Poisson's ratio
	26 - 35	Mass density (used to calculate mass matrix)
	36 - 45	Weight density (used to calculate gravity loads)

C. Element Property Cards (I5,6F10.0)

Columns	1 - 5	Geometric property number
	6 - 15	Axial area
	16 - 25	Shear area associated with shear forces in local 2-direction
	26 - 35	Shear area associated with shear forces in local 3-direction
	36 - 45	Torsional inertia
	46 - 55	Flexural inertia about local 2-axis
	56 - 65	Flexural inertia about local 3-axis

or dimensions of cross-section

One card is required for each unique set of properties. Shear areas need be specified only if shear deformations are to be included in the analysis.



LOCAL COORDINATE SYSTEM FOR BEAM ELEMENT

D. Element Load Factors (4F10.0)

Nodal point loads (no moments) due to gravity are computed. Three cards need be supplied which specify the fraction of these loads (in each of the three global coordinate directions) to be added to each element load case.

Card 1: Multiplier of gravity load in the +X direction

Columns	1 - 10	Element load case A
	11 - 20	Element load case B
	21 - 30	Element load case C
	31 - 40	Element load case D

Card 2: As above for gravity in the +Y direction

Card 3: As above for gravity in the +Z direction

E. Fixed-End Forces (I5,6F10.0/I5,6F10.0)

Two cards are required for each unique set of fixed-end forces occurring in the analysis. Distributed loads and thermal loads can be specified using the fixed-end forces.

Card 1:

Columns	1 - 5	Fixed-end force number
	6 - 15	Fixed-end force in local 1-direction at Node I
	16 - 25	Fixed-end force in local 2-direction at Node I
	26 - 35	Fixed-end force in local 3-direction at Node I
	36 - 45	Fixed-end moment about local 1-direction at Node I
	46 - 55	Fixed-end moment about local 2-direction at Node I
	56 - 65	Fixed-end moment about local 3-direction at Node I

IV. ELEMENT DATA (continued)

Card 2:

Columns	1 - 5	Blank
	6 - 15	Fixed-end force in local 1-direction at Node J
	16 - 25	Fixed-end force in local 2-direction at Node J
	26 - 35	Fixed-end force in local 3-direction at Node J
	36 - 45	Fixed-end moment about local 1-direction at Node J
	46 - 55	Fixed-end moment about local 2-direction at Node J
	56 - 65	Fixed-end moment about local 3-direction at Node J

Note that values input are literally fixed-end values. Corrections due to hinges and rollers are performed within the program. Directions 1, 2 and 3 indicate principal directions in the local beam coordinates

F. Beam Data Cards (10I5,2I6,I8)

Columns	1 - 5	Element number	
	6 - 10	Node number I	
	11 - 15	Node number J	
	16 - 20	Node number K - see accompanying figure	
	21 - 25	Material property number	
	26 - 30	Element property number	
	31 - 35	A	Fixed-end force identification for element load cases A, B, C, and D respectively
	36 - 40	B	
	41 - 45	C	
	46 - 50	D	
	51 - 56	End release code at node I	
	57 - 62	End release code at node J	
	63 - 70	Optional parameter k used for automatic generation of element data. This option is described below under a separate heading. If the option is not used, the field is left blank.	

The end release code at each node is a six digit number of ones and/or zeros. The 1st, 2nd, ... 6th digits respectively correspond to the force components R1, R2, R3, M1, M2, M3 at each node.

If any one of the above element end forces is known to be zero (hinge or roller), the digit corresponding to that component is a one.

NOTES/

(1) If a series of elements occurs in which each element number NE_i is one greater than the previous number NE_{i-1}

i.e., $$NE_i = NE_{i-1} + 1$$

only the element data card for the first element in the series need be given as input, provided

IV. ELEMENT DATA (continued)

(1) The end nodal point numbers are $NI_i = NI_{i-1} + k$

$NJ_i = NJ_{i-1} + k$

and the

(2) material property number
(3) element property number
(4) fixed-end force identification numbers for each element load case
(5) element release code
(6) orientation of local 2-axis

are the same for each element in the series.

The value of k, if left blank, is taken to be one. The element data card for the last beam element must always be given.

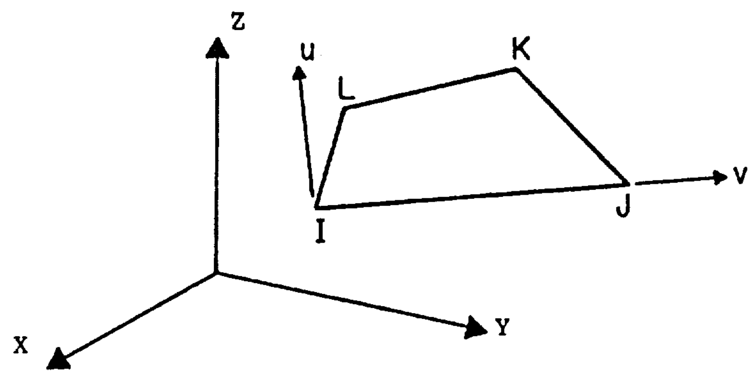
(2) When successive beam elements have the same stiffness, orientation and element loading, the program automatically skips recomputation of the stiffness. Note this when numbering the beams to obtain maximum efficiency.

IV. ELEMENT DATA (continued)

TYPE 3 - PLANE STRESS MEMBRANE ELEMENTS

Quadrilateral (and triangular) elements can be used for plane stress membrane elements of specified thickness which are oriented in an arbitrary plane. All elements have temperature-dependent orthotropic material properties. Incompatible displacement modes can be included at the element level in order to improve the bending properties of the elements.

A general quadrilateral element is shown below:



A local element coordinate system is defined by a u-v system. The v-axis coincides with the I-J side of the element. The u axis is normal to the v-axis and is in the plane defined by nodal points I, J and L. Node K must be in the same plane if the element stiffness calculations are to be correct. The following sequence of cards define the input data for a set of TYPE 3 elements.

A. Control Card (6I5)

Columns 1 - 5	The number 3
6 - 10	Total number of plane stress elements
11 - 15	Number of material property cards
16 - 20	Maximum number of temperature points for any one material; see Section B below.
30	Non-zero numerical punch will suppress the introduction of incompatible displacement modes.

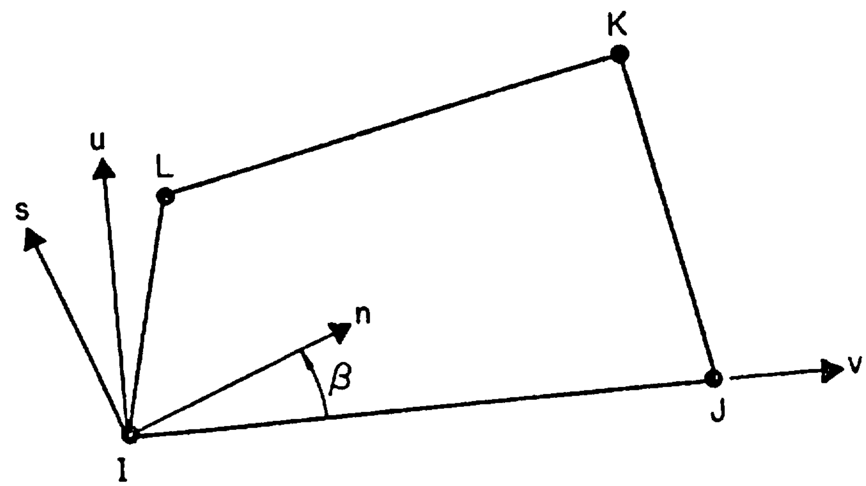
B. Material Property Information

Orthotropic, temperature-dependent material properties are possible. For each different material, the following group of cards must be supplied.

IV. ELEMENT DATA (continued)

1. Material Property Card (2I5,3F10.0)

Columns	1 - 5	Material identification number
	6 - 10	Number of different temperatures for which properties are given. If this field is left blank, the number is taken as one.
	11 - 20	Weight density of material (used to calculate gravity loads)
	21 - 30	Mass density (used to calculate mass matrix)
	31 - 40	Angle β in degrees, measured counter-clockwise from the v-axis to the n-axis.



The n-s axes are the principal axes for the orthotropic material. Weight and mass densities need be listed only if gravity and inertia loads are to be considered.

2. Two cards for each temperature:

Card 1: (8F10.0)

Columns	1 - 10	Temperature
	11 - 20	Modulus of Elasticity - E_n
	21 - 30	Modulus of Elasticity - E_s
	31 - 40	Modulus of Elasticity E_t
	41 - 50	Strain Ratio - ν_{ns}
	51 - 60	Strain Ratio - ν_{nt}
	61 - 70	Strain Ratio - ν_{st}
	71 - 80	Shear Modulus - G_{ns}

IV. ELEMENT DATA (continued)

Card 2: (3F10.0)

Columns	1 - 10	Coefficient of thermal expansion - α_n
	11 - 20	Coefficient of thermal expansion - α_s
	21 - 30	Coefficient of thermal expansion - α_t

All material constants must always be specified. For plane stress, the program modifies the constitutive relations to satisfy the condition that the normal stress σ_t equals zero.

C. Element Load Factors (5F10.0)

Four cards are used to define the element load cases A, B, C and D as fraction of the basic thermal, pressure and acceleration loads.

First card, load case A: Second card, load case B, etc.

Columns	1 - 10	Fraction of thermal load
	11 - 20	Fraction of pressure load
	21 - 30	Fraction of gravity in X-direction
	31 - 40	Fraction of gravity in Y-direction
	41 - 50	Fraction of gravity in Z-direction

D. Element Cards (6I5,2F10.0,2I5,F10.0)

One card per element must be supplied (or generated) with the following information:

Columns	1 - 5	Element number
	6 - 10	Node I
	11 - 15	Node J
	16 - 20	Node K
	21 - 25	Node L (Node L must equal Node K for triangular elements)
	26 - 30	Material identification number
	31 - 40	Reference temperature for zero stresses within element
	41 - 50	Normal pressure on I-J side of element
	51 - 55	Stress evaluation option "n"
	56 - 60	Element data generator "k"
	61 - 70	Element thickness

NOTES/

(1) Element Data Generation - Element cards must be in element number sequence. If cards are omitted, data for the omitted elements will be generated. The nodal numbers will be generated with respect to the first card in the series as follows:

$$I_n = I_{n-1} + k$$

$$J_n = J_{n-1} + k$$

$$K_n = K_{n-1} + k$$

$$L_n = L_{n-1} + k$$

All other element information will be set equal to the information on the last card read. The data generation parameter "k" is specified on that card.

(2) Stress Print Option - See element type 4

(3) Thermal Data - See element type 4

(4) Use of Triangles - See element type 4

(5) Use of Incompatible Modes - See element type 4

IV. ELEMENT DATA (continued)

TYPE 4 - TWO-DIMENSIONAL FINITE ELEMENTS

Quadrilateral (and triangular) elements can be used as:

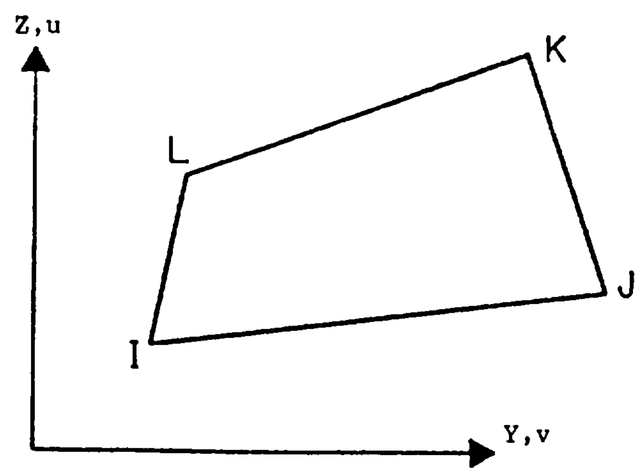
(i) Axisymmetric solid elements symmetrical about the Z-axis. The radial direction is specified as the Y-axis. Care must be exercised in combining this element with other types of elements.

(ii) Plane strain elements of unit thickness in the Y-Z plane.

(iii) Plane stress elements of specified thickness in the Y-Z plane.

All elements have temperature-dependent orthotropic material properties. Incompatible displacement modes can be included at the element level in order to improve the bending properties of the element.

A general quadrilateral element is shown below:



A. Control Card (6I5)

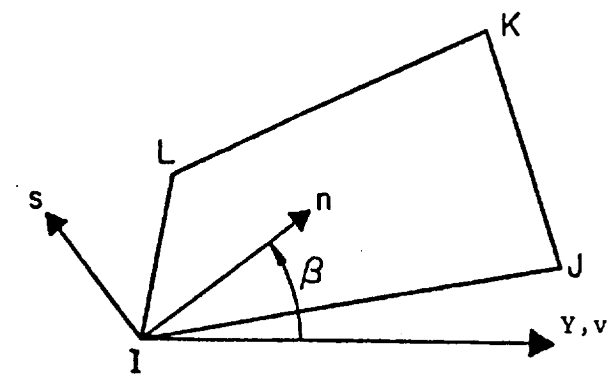
Columns	1 - 5	The number 4
	6 - 10	Total number of elements
	11 - 15	Number of different materials
	16 - 20	Maximum number of temperature cards for any one material - see Section B below.
	25	0 for axisymmetric analysis 1 for plane strain analysis 2 for plane stress analysis
	30	Non-zero numerical punch will suppress the introduction of incompatible displacement modes. Incompatible modes cannot be used for triangular elements and are automatically suppressed.

B. Material Property Information

Orthotropic, temperature-dependent material properties are possible. For each different material the following group of cards must be supplied.

1. Material Property Card (2I5,3F10.0)

Columns	1 - 5	Material identification number
	6 - 10	Number of different temperature for which properties are given. If this field is left blank, the number is taken as one.
	11 - 20	Weight density of material (used to calculate gravity loads)
	21 - 30	Mass density (used to calculate mass matrix)
	31 - 40	Angle β in degrees, measured counterclockwise from the v-axis to the n-axis.



PRINCIPAL MATERIAL AXES

The n-s axes are the principal axes for the orthotropic material. Weight density is needed only if gravity and inertia loads are to be considered.

2. Two cards for each temperature:

Card 1: (8F10.0)

Columns	1 - 10	Temperature
	11 - 20	Modulus of elasticity - E_n
	21 - 30	Modulus of elasticity - E_s
	31 - 40	Modulus of elasticity - E_t
	41 - 50	Strain ratio - ν_{ns}
	51 - 60	Strain ratio - ν_{nt}
	61 - 70	Strain ratio - ν_{st}
	71 - 80	Shear modulus - G_{ns}

IV. ELEMENT DATA (continued)

Card 2: (3F10.0)

Columns	1 - 10	Coefficient of thermal expansion - α_n
	11 - 20	Coefficient of thermal expansion - α_s
	21 - 30	Coefficient of thermal expansion - α_t

All material constants must always be specified. In plane stress, the program modifies the constitutive relations to satisfy the condition that the normal stress σ_t equals zero.

C. Element Load Factors

Four cards are used to define the element load cases A, B, C and D as fraction of the basic thermal, pressure and acceleration loads.

First card, load case A; Second card, load case B; etc.

Columns	1 - 10	Fraction of thermal load
	11 - 20	Fraction of pressure load
	21 - 30	Fraction of gravity in X-direction
	31 - 40	Fraction of gravity in Y-direction
	41 - 50	Fraction of gravity in Z-direction

D. Element Cards (6I5,2F10.0,2I5,F10.0)

One card per element must be supplied (or generated) with the following information:

Columns	1 - 5	Element number
	6 - 10	Node I
	11 - 15	Node J
	16 - 20	Node K
	21 - 25	Node L (Node L must equal Node K for triangular elements)
	26 - 30	Material identification number
	31 - 40	Reference temperature for zero stresses within element
	41 - 50	Normal pressure on I-J side of element
	51 - 55	Stress evaluation option "n"
	56 - 60	Element data generator "k"
	61 - 70	Element thickness (For plane strain set equal to 1.0 by program)

NOTES/

(1) Element Data Generation - Element cards must be in element number sequence. If cards are omitted the omitted element data will be generated. The nodal numbers will be generated with respect to the first card in the series as follows:

$$I_n = I_{n-1} + k$$

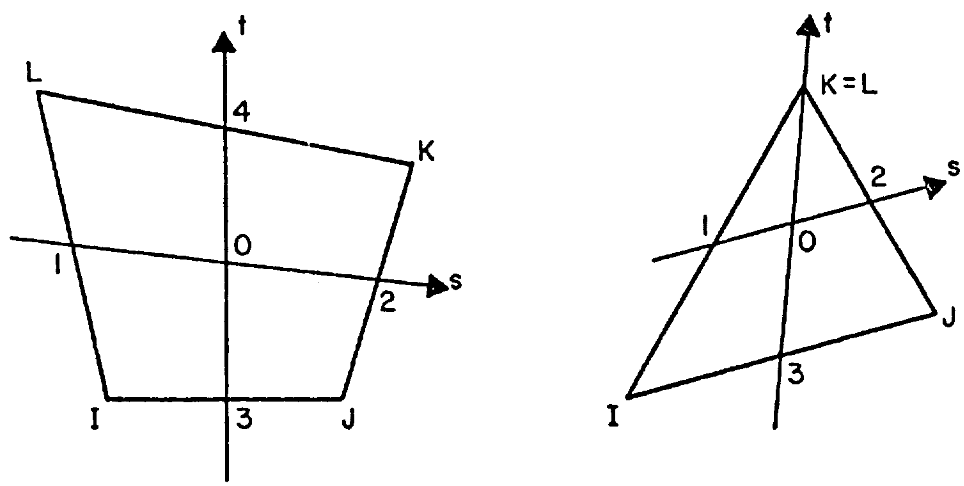
$$J_n = J_{n-1} + k$$

$$K_n = K_{n-1} + k$$

$$L_n = L_{n-1} + k$$

All other element information will be set equal to the information on the last card read. The data generation parameter k is given on that card.

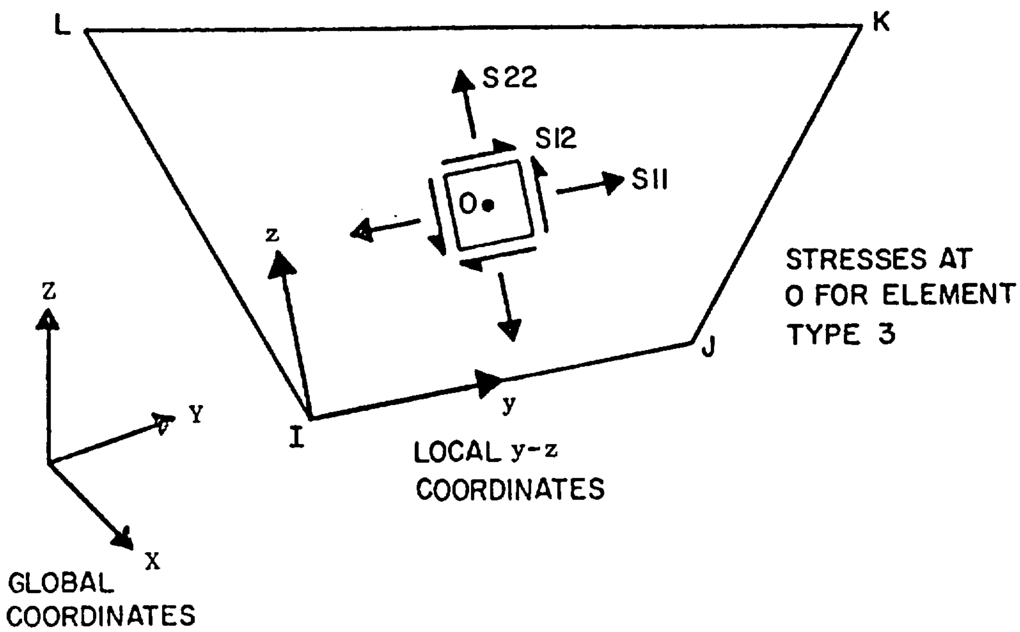
(2) Stress Print Option - The following description of the stress print option applies to both element types 3 and 4. The value of the stress print option "n" can be given as 1, 0, 8, 16 or 20.

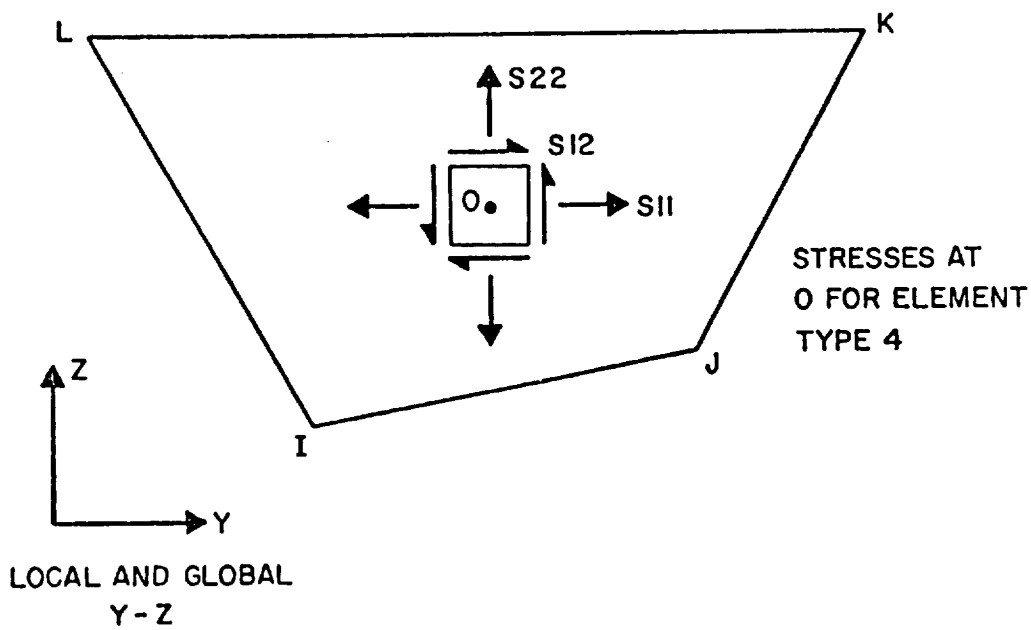


0 = origin of natural s-t coordinates (Fig. 5-2). Points 1, 2, 3 and 4 are midpoints of sides. The points at which stresses are output depend on the value of n as described in the following table.

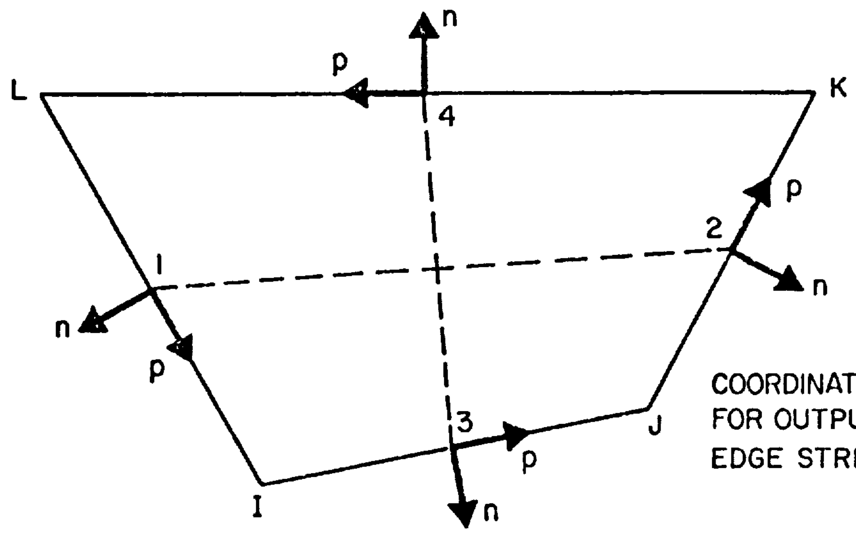
n	Stresses output at
1	None
0	0
8	0, 1
16	0, 1, 2, 3
20	0, 1, 2, 3, 4

The stresses at O are printed in a local y-z coordinate system. For element type 3, side I-J defines the local y-z axes in the plane of the element. For element type 4 the local y-z axes are parallel to the global Y-Z axes.

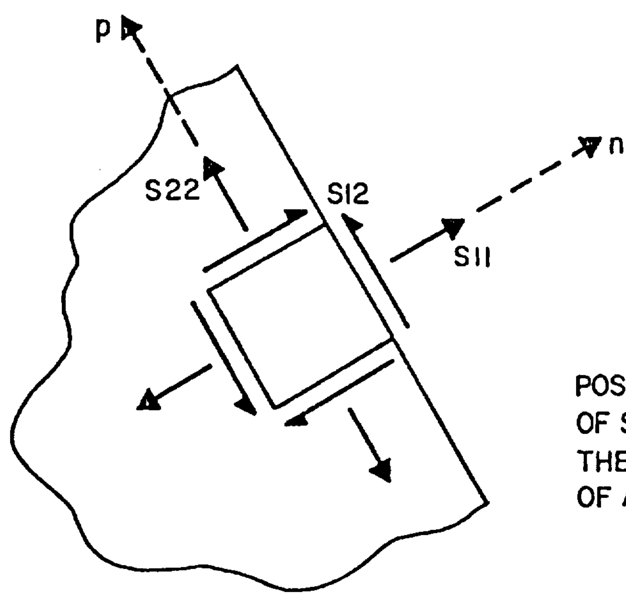




For both element types 3 and 4 the stresses at each edge midpoint are output in a rectangular n-p coordinate system defined by the outward normal to the edge (n axis) and the edge (p axis). The positive p axis for points 1, 2, 3 and 4 is from L to I, J to K, I to J and K to L respectively (positive direction is counterclockwise about element).



COORDINATE SYSTEMS FOR OUTPUT OF EDGE STRESSES



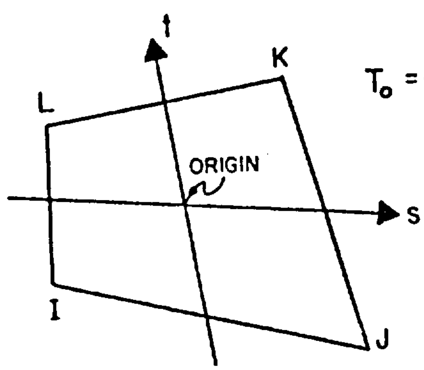
POSITIVE STATE OF STRESS AT THE MIDPOINT OF A SIDE

The stresses for an element are output under the following headings: S11, S22, S12, S33, S-MAX, S-MIN, ANGLE. The normal stresses S11 and S22 and the shear stress S12 are as described above. S-MAX and S-MIN are the principal stresses in the plane of the element and S33 is the third principal stress acting on the plane of the element. ANGLE is the angle in degrees from (1) the local y axis at point 0, or (2) the n axis at the midpoints, to the axis of the algebraically largest principal stress.

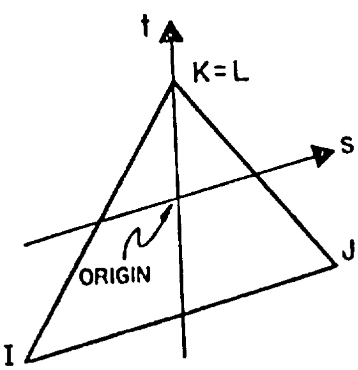
For triangular elements the stress print option is as described above except that n = 20 is not valid. If n = 20 is input, n will be set to 16 by the program.

(3) Thermal Data - Nodal temperatures as specified on the nodal point data cards are used by element types 3 and 4 in the following two ways:

(1) Temperature-dependent material properties are approximated by interpolating (or extrapolating) the input material properties at the temperature T_o corresponding to the origin of the local s-t coordinate system (see Fig. 5.2 for description of local element coordinates). The material properties throughout the element are assumed constant corresponding to this temperature.



$$T_o = \frac{T_I + T_J + T_K + T_L}{4.0}$$



(2) For computation of nodal loads due to thermal strains in the element a bilinear interpolation expansion for the temperature change $\Delta T\ (s,t)$ is used.

$$\Delta T\ (s,t) = \sum_{i=1}^{4} h_i(s,t)\ T_i - T_r$$

where T_i are the nodal temperatures specified on the joint data cards, T_r is the reference stress free temperature and $h_i\ (s,t)$ are the interpolation functions given by Eq. 5.7.

(4) Use of Triangles - In general, the elements are most effective when they are rectangular, i.e. the elements are not distorted. Therefore, regular and rectangular element mesh layouts should be used as much as possible. In particular, the triangle used is the constant strain triangle; and it should be avoided, since its accuracy is not satisfactory.

(5) Use of Incompatible Modes - Incompatible displacement modes have been found to be effective only when used in rectangular elements. They should always be employed with care. Since incompatible modes are used for all elements of a group it is recommended to use separate element groups for elements with incompatible modes and elements without incompatible modes, respectively. (See Section II, note (2)).

IV. ELEMENT DATA (continued)

TYPE 5 - THREE-DIMENSIONAL SOLID ELEMENTS (EIGHT NODE BRICK)

General three-dimensional, eight-node, isoparametric elements with three translational degrees of freedom per node are identified by the number 5. Isotropic material properties are assumed. The element load cases (A, B, C and D) are defined as a combination of surface pressure, hydrostatic loads, inertia loads in three directions and thermal loads. The six components of stress and three principal stresses are computed at the center of each element. Also, surface stresses are evaluated. Nine incompatible displacement modes are assumed in the formation of element stiffnes matrices. For 8-node elements without incompatible modes use element type 8.

A. Control Card (4I5)

Columns	1 - 5	The number 5
	6 - 10	Number of 8-node solid elements
	11 - 15	Number of different materials
	16 - 20	Number of element distributed load sets

B. Material Property Cards (I5,4F10.0) One card for each different material

Columns	1 - 5	Material identification number
	6 - 15	Modulus of elasticity (only elastic, isotropic materials are considered)
	16 - 25	Poisson's ratio
	26 - 35	Weight density of material (for calculation of gravity loads or mass matrix)
	36 - 45	Coefficient of thermal expansion

C. Distributed Surface Loads (2I5,2F10.2,I5) One card is required for each unique set of uniformly distributed surface loads and for each reference fluid level for hydrostatically varying pressure loads. See notes (4) and (5) for sign convention.

Columns	1 - 5	Load set identification number
	6 - 10	LT (load type) LT = 1 if this card specifies a uniformly distributed load. LT = 2 if this card specifies a hydrostatically varying pressure.
	11 - 20	P If LT = 1, P is the magnitude of the uniformly distributed load If LT = 2, P is the weight density of the fluid causing the hydrostatic pressure
	21 - 30	Y If LT = 1, leave blank If LT = 2, Y is the global Y coordinate of the surface of fluid causing hydrostatic pressure loading
	31 - 35	Element face number on which surface load acts. Face numbers are from 1 to 6 as

described in note (5) for uniformly distributed loads and can be only faces 2, 4 or 6 for hydrostatically varying pressures.

D. Acceleration due to gravity (F10.2)

Columns 1 - 10 Acceleration due to gravity (for calculation of mass matrix)

E. Element Load Case Multipliers (5 cards of 4F10.2)

Multipliers on the element load cases are scaling factors in order to provide flexibility in modifying applied loads.

Card 1: Columns	1 - 10	PA	Pressure load multipliers
	11 - 20	PB	
	21 - 30	PC	
	31 - 40	PD	

PA is a factor used to scale the complete set of distributed surface loads. This scaled set of loads is assigned to element load case A. Note that zero is a valid multiplier. PB, PC and PD are similar to PA except that scaled loads are assigned to element load cases B, C and D respectively. For the majority of applications these factors should be 1.0

Card 2: Columns	1 - 10	TA	Thermal load multipliers
	11 - 20	TB	
	21 - 30	TC	
	31 - 40	TD	

TA is a factor used to scale the complete set of thermal loads. The scaled set of loads are then assigned to element load case A. TB, TC and TD are similar and refer to element load cases B, C and D respectively.

Card 3: Columns	1 - 10	GXA	Gravity load multipliers for + X global direction
	11 - 20	GXB	
	21 - 30	GXC	
	31 - 40	GXD	
Card 4: Columns	1 - 10	GYA	Gravity load multipliers for + Y global direction
	11 - 20	GYB	
	21 - 30	GYC	
	31 - 40	GYD	
Card 5: Columns	1 - 10	GZA	Gravity load multipliers for + Z global direction
	11 - 20	GZB	
	21 - 30	GZC	
	31 - 40	GZD	

Gravity loads are computed from the weight density of the material and from the geometry of the element. GXA is a multiplier which reflects the location of the gravity axis and any load factors used. The program computes the weight of the element, multiplies it by GXA and assigns the resulting loads to the + X direction of element load case A. Consequently GXA is the product of the component of gravity along the + X global axis (from - 1.0 to 1.0) and any desired load factor. GXB, GXC and GXD are similar to GXA and refer to element load cases B, C and D respectively. GYA and GZA refer to the global Y and Z directions respectively.

F. Element Cards (12I5,4I2,2I1,F10.2)

Columns 1 - 5	Element number	
6 - 10	Global node point numbers corresponding to element nodes (See note (3))	1
11 - 15		2
16 - 20		3
21 - 25		4
26 - 30		5
31 - 25		6
36 - 40		7
41 - 45		8
46 - 50	Integration Order	
51 - 55	Material Number	
56 - 60	Generation Parameter (INC)	
61 - 62	LSA	LSA is the distributed surface load set identification number of the distributed load acting on this element to be assigned to element load case A. LSB, LSC and LSD refer to element load cases B, C and D respectively
63 - 64	LSB	
65 - 66	LSC	
67 - 68	LSD	
69 - 70	Face numbers for stress output	
71 - 80	Stress-free element temperature	

NOTES/

(1) Element Generation

1. Element cards must be in ascending order
2. Generation is possible as follows:
   If a series of element cards are omitted,
   a. Nodal point numbers are generated by adding INC to those of the preceding element. (If omitted, INC is set equal to 1.)
   b. Same material properties are used as for the preceding element.
   c. Same temperature is used for succeeding elements.

d. If on first card for the series the integration order is:
   >0 Same value is used for succeeding elements.
   = 0 A new element stiffness is not formed. Element stiffness is assumed to be identical to that of the preceding element.
   <0 Absolute value is used for the first element of the series, and the same element stiffness is used for succeeding elements.

e. If on first card for the series, the distributed load number (for any load case) is:
   >0 Same load is applied to succeeding elements.
   <0 The load case is applied to this element but not to succeeding elements in the series.

3. Element card for the last element must be supplied.

(2) Integration Order

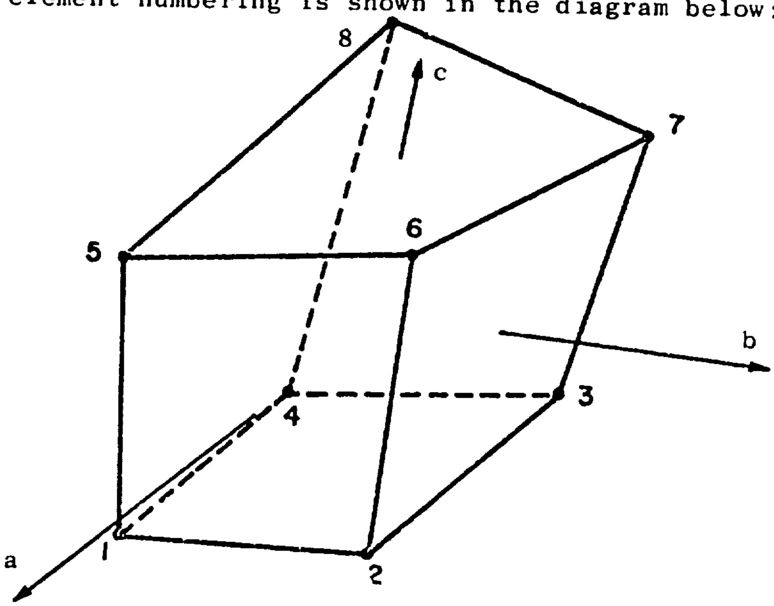
Computation time (for element stiffness) increases with the third power of the integration order. Therefore, the smallest satisfactory order should be used. This is found to be:

2 for rectangular element
3 for skewed element
1 may be used if element is extremely distorted in shape, but not recommended.

Mesh should be selected to give "rectangular" elements as far as possible.

(3) Element Coordinate System

Local element coordinate system is a natural system for this element in which the element maps onto a cube. Local element numbering is shown in the diagram below:



(4) Identification of Element Faces

Element faces are numbered as follows:

Face 1 corresponds to + a direction
2 corresponds to - a direction
3 corresponds to + b direction
4 corresponds to - b direction
5 corresponds to + c direction
6 corresponds to - c direction
0 corresponds to the center of the element

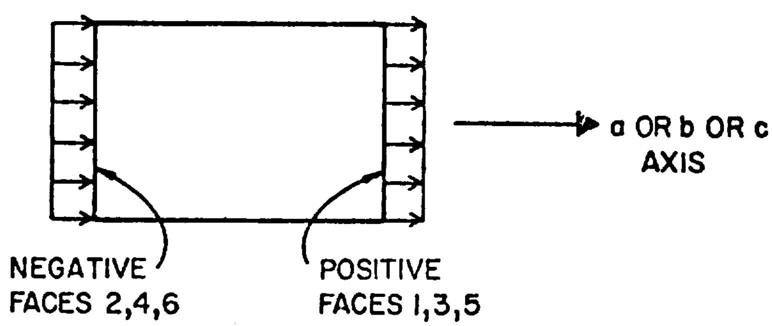
Faces 1,3,5 are positive faces

Faces 2,4,6 are negative faces

(5) Distributed Surface Loads

Two types of surface loadings may be specified; load type 1 (LT = 1), uniformly distributed surface load and load type 2 (LT = 2), hydrostatically varying surface pressure (but not surface tension). Both loading types are for loads normal to the surface and do not include surface shears. Surface loadings that do not fall into these categories must be input as nodal loads on the concentrated load data cards (see Section V).

(1) LT = 1: A positive surface load acts in the direction of the outward normal of a positive element face and along the inward normal of a negative element face as shown in the following diagram.



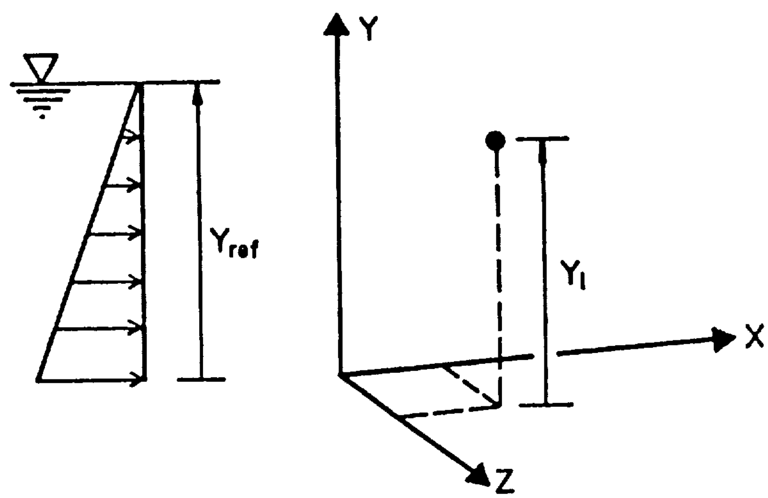
POSITIVE SURFACE LOADING P

If the uniformly distributed surface loading P is input as a positive quantity then it describes pressure loading on faces 2, 4 or 6 and tensile loading on faces 1, 3 or 5. If P is input as a negative quantity then it describes tensile loading on faces 2, 4 or 6 and pressure on faces 1, 3 or 5.

(2) LT = 2: A hydrostatically varying surface pressure on element faces 2, 4 or 6 can be specified by a reference fluid surface and a fluid weight density γ as input. Only one hydrostatic surface pressure card need be input in order to specify a hydrostatic loading on the complete structure. The consistent nodal loads are calculated by the program as follows. At each numerical integration point "i" on an element surface the pressure P_i is calculated from

$$P_i = \gamma (Y_i - Y_{ref})$$

where Y_i is the global Y coordinate of the point in question and Y_{ref} specifies the fluid surface assuming gravity acts along the -Y axis



If $P_i > 0$, corresponding to surface tension, the contribution is ignored. If an element face is such that $Y_i > Y_{ref}$ for all i (16 integration points are used by program) then no nodal loads will be applied to the element. If some $P_i > 0$ and some $P_i < 0$ for a particular face, then approximate nodal loads are obtained for the partially loaded surface.

(6). Thermal Loads

Thermal loads are computed assuming a constant temperature increase ΔT throughout the element.

ΔT = $T_{avg} - T_o$

T_{avg} = the average of the 8 nodal point temperatures specified on nodal point data cards

T_o = stress free element temperature specified on the element card.

(7). Element Load Cases

Element load case A consists of all the contributions from distributed loadings, thermal loadings and gravity loading for all the elements taken collectively.

Load case A = Σ (PA x pressure loading
+ TA x thermal loading
+ GXA x gravity X loading
+ GYA x gravity Y loading
+ GZA x gravity Z loading)

Element load case A for the set of three dimensional solid elements is added to element load case A for the other element types in the analysis. The treatment of element load cases B, C and D is analogous to that of element load case A. The loading cases for the structure are obtained by adding linear combinations of element load cases A, B, C and D to the nodal loads specified on the joint data cards.

(8) Output of Element Stresses

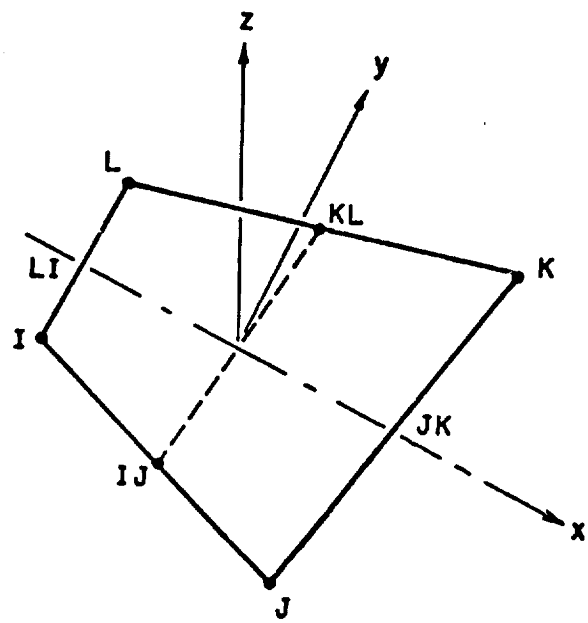
1. At the centroid of the element, stresses are referred to the global axes. Three principal stresses are also presented.

2. At the center of an element face, stresses are referred to a set of local axes (x,y,z). These local axes are individually defined for each face as follows: Let nodal points I, J, K and L be the four corners of the element face. Then

   x is specified by LI - JK, where LI and JK are midpoints of sides L-I and J-K,

   z is normal to x and to the line joining midpoints IJ and KL.

   y is normal to x and z, to complete the right-handed system.



The corresponding nodal points I, J, K and L in each face are given in the table.

FACE	NODAL POINTS			
	I	J	K	L
1	1	2	6	5
2	4	3	7	8
3	3	7	6	2
4	4	8	5	1
5	8	5	6	7
6	4	1	2	3

Two surface principal stresses and the angle between the algebraically largest principal stress and the local x axis are printed with the output. It is optional to choose one or two locations of an element where stresses are to be computed. In the output, "face zero" designates the centroid of the element.

IV. ELEMENT DATA (continued)

TYPE 6 - PLATE AND SHELL ELEMENTS (QUADRILATERAL)

A. Control Card (3I5)

Columns	1 - 5	The number 6
	6 - 10	Number of shell elements
	11 - 15	Number of different materials

B. Material Property Information

Anisotropic material properties are possible. For each different material, two cards must be supplied.

Card 1: (I10,20X,4F10.0)

Columns	1 - 10	Material identification number
	31 - 40	Mass density
	41 - 50	Thermal expansion coefficient α_x
	51 - 60	Thermal expansion coefficient α_y
	61 - 70	Thermal expansion coefficient α_{xy}

Card 2: (6F10.0)

Columns	1 - 10	Elasticity element C_{xx}
	11 - 20	Elasticity element C_{xy}
	21 - 30	Elasticity element C_{xs}
	31 - 40	Elasticity element C_{yy}
	41 - 50	Elasticity element C_{ys}
	51 - 60	Elasticity element G_{xy}

Elements in plane stress material matrix [C]

$$\begin{Bmatrix} \sigma_{xx} \\ \sigma_{yy} \\ \tau_{xs} \end{Bmatrix} = \begin{bmatrix} C_{xx} & C_{xy} & C_{xs} \\ C_{xy} & C_{yy} & C_{ys} \\ C_{xs} & C_{ys} & G_{xy} \end{bmatrix} \begin{Bmatrix} \epsilon_{xx} \\ \epsilon_{yy} \\ \gamma_{xy} \end{Bmatrix}$$

C. Element Load Multipliers (5 cards)

Card 1: (4F10.0)

Columns	1 - 10	Distributed lateral load multiplier for load case A
	11 - 20	Distributed lateral load multiplier for load case B
	21 - 30	Distributed lateral load multiplier for load case C
	31 - 40	Distributed lateral load multiplier for load case D

Card 2: (4F10.0)

Columns	1 - 10	Temperature multiplier for load case A
	11 - 20	Temperature multiplier for load case B
	21 - 30	Temperature multiplier for load case C
	31 - 40	Temperature multiplier for load case D

Card 3: (4F10.0)

Columns	1 - 10	X-direction acceleration for load case A
	11 - 20	X-direction acceleration for load case B
	21 - 30	X-direction acceleration for load case C
	31 - 40	X-direction acceleration for load case D

IV. ELEMENT DATA (continued)

Card 4: (4F10.0) Same as Card 3 for Y-direction

Card 5: (4F10.0) Same as Card 3 for Z-direction

D. Element Cards (8I5,F10.0)

One card for each element

Columns		
Columns	1 - 5	Element number
	6 - 10	Node I
	11 - 15	Node J
	16 - 20	Node K
	21 - 25	Node L
	26 - 30	Node 0
	31 - 35	Material identification (if left blank, taken as one)
	36 - 40	Element data generator K_n
	41 - 50	Element thickness
	51 - 60	Distributed lateral load (pressure)
	61 - 70	Mean temperature variation T from the reference level in undeformed position
	71 - 80	Mean temperature gradient $\partial T/\partial z$ across the shell thickness (a positive temperature gradient produces a negative curvature).

NOTES/

(1) Nodal Points and Coordinate Systems

The nodal point numbers I, J, K and L are in sequence in a counter-clockwise direction around the element. The local element coordinate system (x, y, z) is defined as follows:

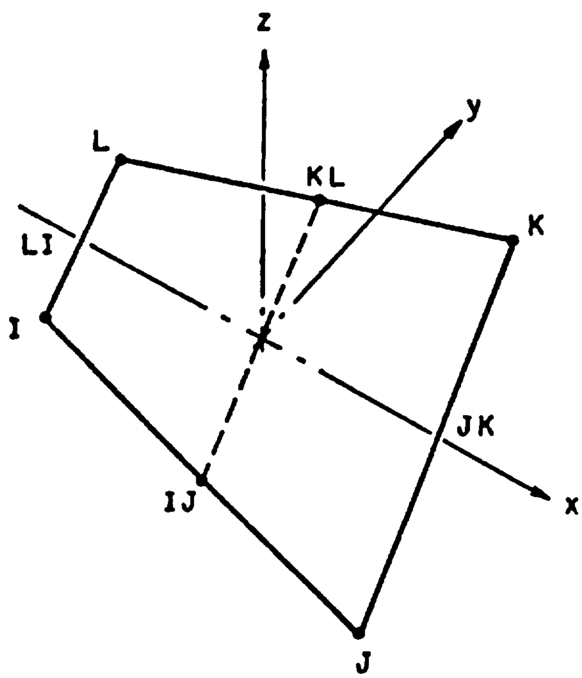
x Specified by LI - JK, where LI and JK are midpoints of sides L-I and J-K.

z Normal to x and to the line joining midpoints IJ and KL.

y Normal to x and z to complete the right-handed system.

This system is used to express all physical and kinematic shell properties (stresses, strains, material law, etc.), except that the body force density is referred to the global coordinate system (X, Y, Z).

IV. ELEMENT DATA (continued)



For the analyses of shallow shells, rotational constraints normal to the surface may be imposed by the addition of boundary elements at the nodes (element type #7).

(2) Node 0

When columns 26 - 30 are left blank, mid-node properties are computed by averaging the four nodes.

(3) Element Data Generation

Element cards must be in element number sequence. If element cards are omitted, the program automatically generates the omitted information as follows:

The increment for element number is one

$$\text{i.e.} \quad NE_{i+1} = NE_i + 1$$

The corresponding increment for nodal number is K_n

$$\text{i.e.} \quad NI_{i+1} = NI_i + K_n$$

$$NJ_{i+1} = NJ_i + K_n$$

$$NK_{i+1} = NK_i + K_n$$

$$NL_{i+1} = NL_i + K_n$$

Material identification, element thickness, distributed lateral load, temperature and temperature gradient for generated elements are the same. Always include the complete last element card.

IV. ELEMENT DATA (continued)

(4) Element Stress Calculations

Output are moments per unit length and membrane stresses.

IV. ELEMENT DATA (continued)

TYPE 7 - BOUNDARY ELEMENTS

This element is used to constrain nodal displacements to specified values, to compute support reactions and to provide linear elastic supports to nodes. If the boundary condition code for a particular degree of freedom is specified as 1 on the structure nodal point data cards, the displacement corresponding to that degree of freedom is zero and no support reactions are obtained with the printout. Alternatively, a boundary element can be used to accomplish the same effect except that support reactions are obtained since they are equal to the member end forces of the boundary elements which are printed. In addition the boundary element can be used to specify non-zero nodal displacements in any direction which is not possible using the nodal point data cards.

The boundary element is defined by a single directed axis through a specified nodal point, by a linear extensional stiffness along the axis or by a linear rotational stiffness about the axis. The boundary element is essentially a spring which can have axial displacement stiffness and axial rotational stiffness. There is no limit to the number of boundary elements which can be applied to any joint to produce the desired effects. Boundary elements have no effect on the size of the stiffness matrix.

INPUT DATA

A. Control Card (2I5)

Columns	1 - 5	The number 7.
	6 - 10	Total number of boundary elements.

B. Element Load Multipliers (4F10.0)

Columns	1 - 10	Multiplier for load case A
	11 - 20	Multiplier for load case B
	21 - 30	Multiplier for load case C
	31 - 40	Multiplier for load case D

C. Element Cards (8I5,3F10.0)

One card per element (in ascending nodal point order) except where automatic element generation is used.

Columns	1 - 5	Node N, at which the element is placed
	6 - 10	Node I
	11 - 15	Node J — Leave columns 11 - 25 blank if only node I is needed.
	16 - 20	Node K
	21 - 25	Node L
	26 - 30	Code for displacement
	31 - 35	Code for rotation
	36 - 40	Data generator K_n
	41 - 50	Specified displacement along element axis
	51 - 60	Specified rotation about element axis
	61 - 70	Spring stiffness (set to 10^{10} if left blank) for both extension and rotation.

IV. ELEMENT DATA (continued)

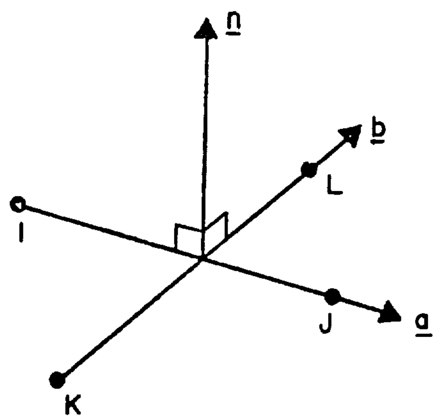
NOTES/

(1) Direction of boundary element

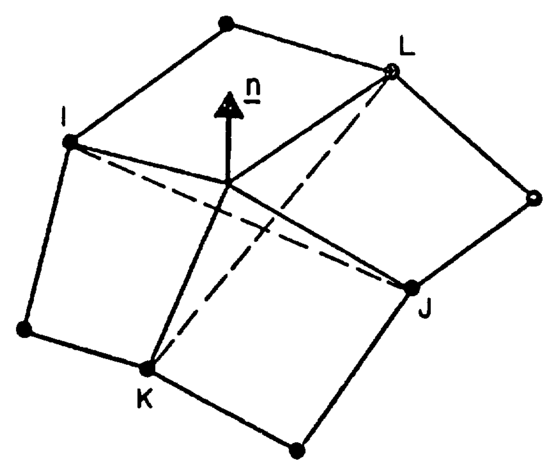
The direction of the boundary element at node N is specified in one of two ways.

(i) A second nodal point I defines the direction of the element from node N to node I.

(ii) Four nodal points I, J, K and L specify the direction of the element as the normal to the plane defined by two intersecting straight lines (vectors $\underline{a}$ and $\underline{b}$, see Fig. below).



$\underline{n} = \underline{a} \times \underline{b}$



ROTATIONAL CONSTRAINT
IN THIN SHELL ANALYSIS

The four points I, J, K and L need not be unique. A useful application for the analysis of shallow thin shells employs the boundary element to approximate rotational constraint about the surface normal as shown above.

$\underline{n}$ is given by the vector cross product $\underline{n} = \underline{a} \times \underline{b}$ and defines the direction of the boundary element.

Note that node I in case (i) and nodes I, J, K and L in case (ii) are used only to define the direction of the element and if convenient may be any nodes used to define other elements. However 'artificial nodes' may be created to define directions of boundary elements. These 'artificial nodes' are input on the nodal point data cards with their coordinates and with all the boundary condition codes specified as 1 (one).

It should be noted that node N is the structure node to which the boundary element is attached. In case (i), a positive displacement moves node N towards node I. Correspondingly, a positive force in the element means compression in the element. In case (ii), a positive displacement moves node N into the direction $\underline{n}$ (see Fig.).

(2) Displacement and rotation codes

Displacement code = 1: When this code is used, the displacement δ, specified in columns 41-50, and the spring stiffness k, specified in columns 61-70, are used by the program in the following way. The load P, evaluated from $P = k\delta$, is applied to node N in the direction node N to node I in case (i) and into direction $\underline{n}$ in case (ii), if δ is positive. If k is much greater than the stiffness of the structure at node N without the boundary element, then the net effect is to produce a displacement very nearly equal to δ at node N. If $\delta = 0$, then $P = 0$ and the stiff spring approximates a rigid support. Note that the load P will contribute to the support reaction for nonzero δ. The boundary condition codes specified on the structure nodal point data cards must be consistent with the fact that a load P is being applied to node N to effect the desired displacement (even when this displacement is zero).

Rotation code = 1: This case is analogous to the situation described above. A torque T, evaluated from $T = k\ \theta$, is applied to node N about the axis (direction) of the element. The rotation θ is specified in columns 51-60.

(3) Data generator K_n

When a series of nodes are such that:

(i) All have identical boundary elements attached
(ii) All boundary elements have same direction
(iii) All specified displacements and rotations are identical
(iv) The nodal sequence forms an arithmetic sequence, i.e., N, $N + K_n$, $N + 2K_n$ etc.,

then only the first and last node in the sequence need be input. The increment K_n is input in columns 36-40 of the first card.

(4) Element load multipliers

Each of the four possible element load cases A, B, C and D associated with the boundary elements consists of the complete set of displacements as specified on the boundary element cards multiplied by the element load multiplier for the corresponding load case. As an example, suppose that displacement of node N is specified as 1.0, spring stiffness as 10^{10} and no other boundary element displacements are specified. Let case A multiplier be 0.0 and case B multiplier be 2.0. For element load case A the specified displacement is $0.0 \times 1.0 = 0.0$ while that for B is $2.0 \times 1.0 = 2.0$. Linear combinations of element load cases A, B, C and D for all types of elements collectively for a particular problem are specified on the structure element load multiplier cards. As far as the boundary element is concerned, this device is useful when a particular node has a support displacement in one load case but is fixed in others.

(5) Recommendations for use of boundary elements

If a boundary element is aligned with a global displacement direction, only the corresponding diagonal element in the stiffness matrix is modified. Therefore, no stiffness matrix ill-conditioning results. However, when the boundary element couples degrees of freedom, large off-diagonal elements introduce ill-conditioning into the stiffness matrix which can cause solution difficulties.

In the analysis of shallow shells boundary elements with stiffness a fraction of the element bending stiffness should be used (say less than or about 10%).

In dynamic analysis "artificially stiff" boundary elements should not be used. (See note (8) in Section VII.A).

TYPE 8 - VARIABLE-NUMBER-NODES THICK SHELL AND THREE-DIMENSIONAL ELEMENTS

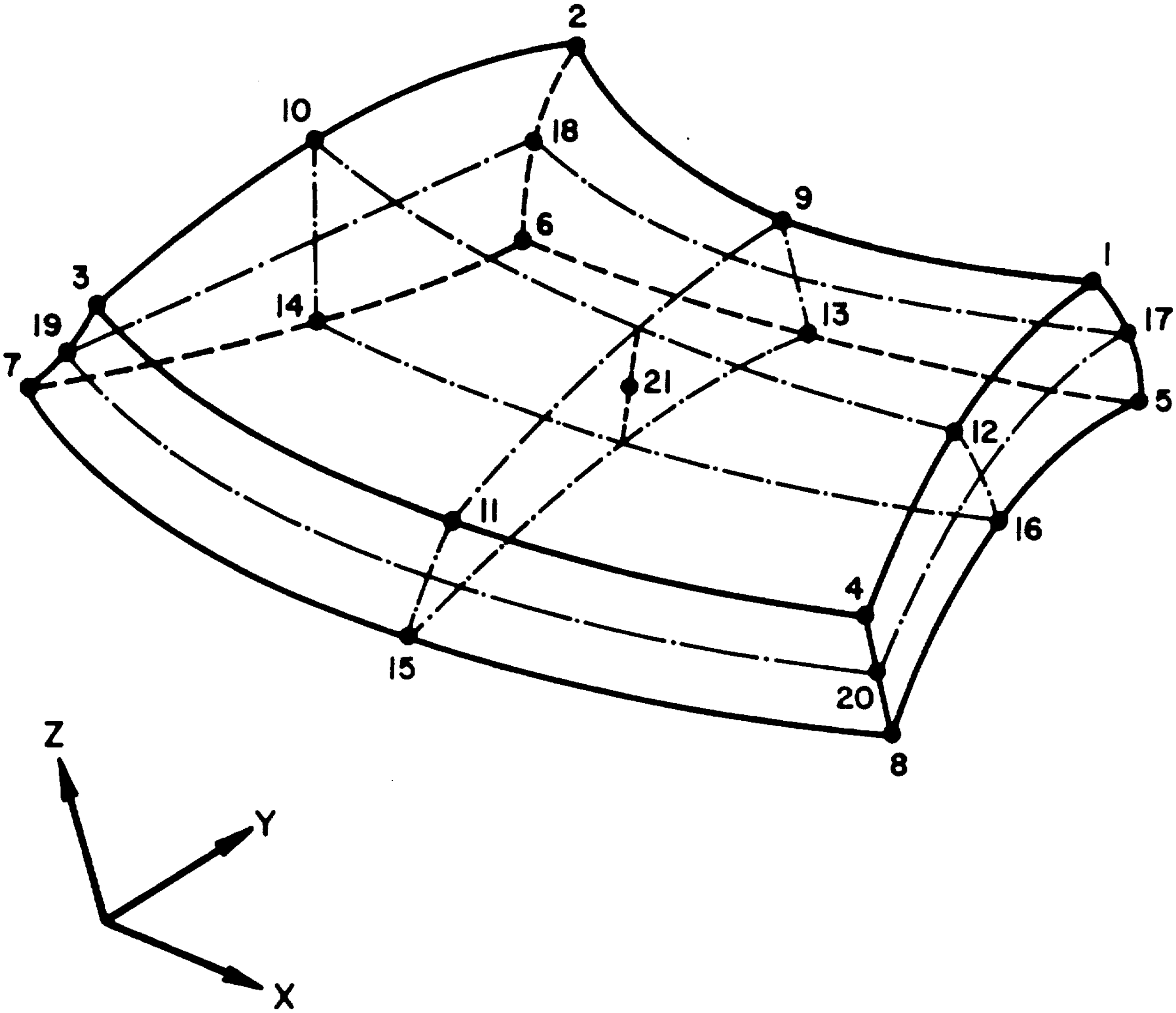
A minimum of 8 and a maximum of 21 nodes are used to describe a general three dimensional isoparametric element; the element is used to represent orthotropic, elastic media. The element type is identified by the number eight (8). Three translational degrees of freedom are assigned to each node, and at least the eight corner nodes must be input to define a hexahedron. Input of nodes 9 to 21 is optional; the figures below illustrate some of the most commonly used node combinations.

Element load cases (A,B,C,...) are formed from combinations of applied surface pressure, hydrostatic loads, inertia loads in the three directions X,Y,Z and thermal loads. Six global stresses are output at up to seven (7) locations within the element; these output locations are selected by means of appropriate data entries.

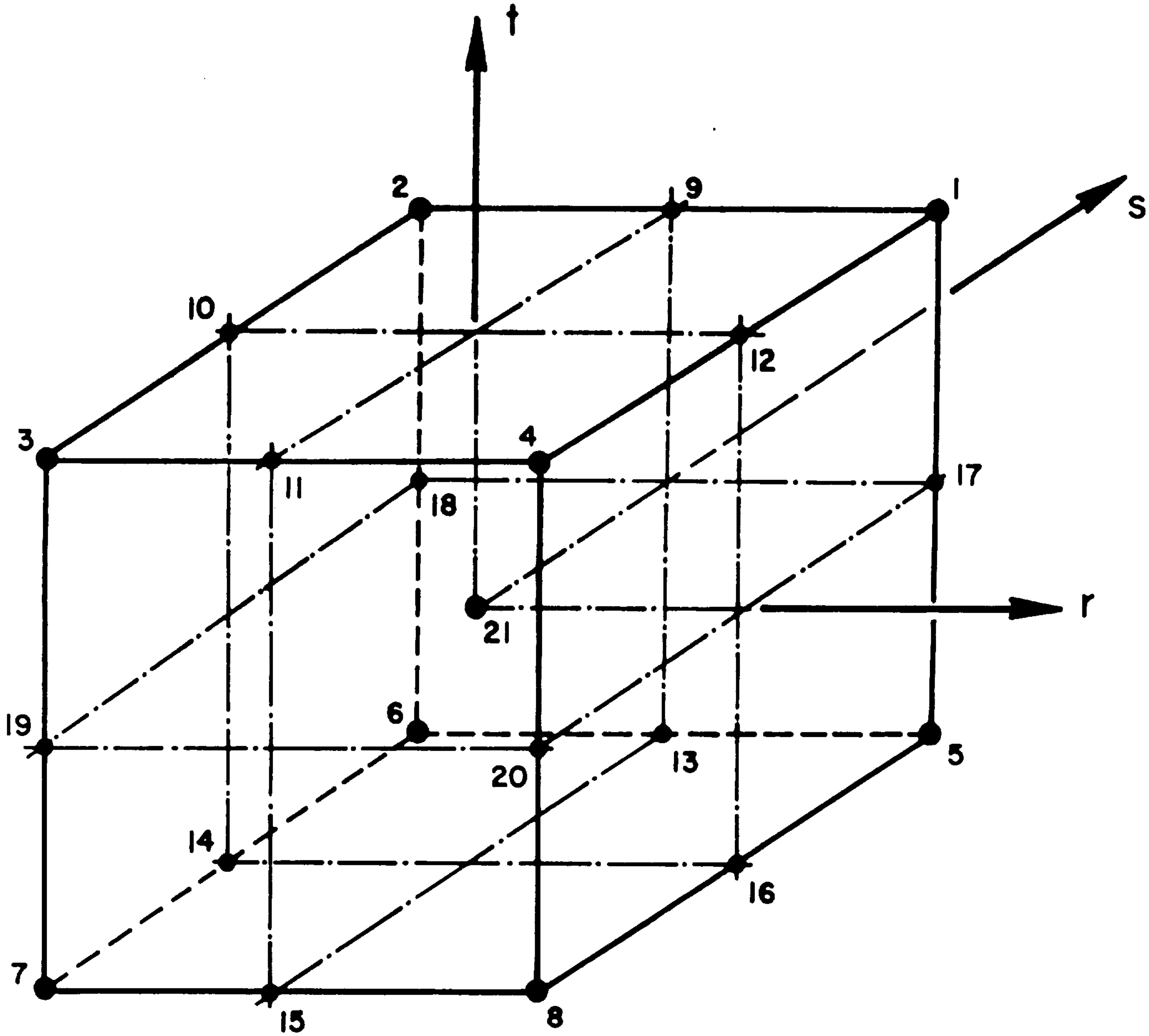
Node temperatures input in Section III are used to form an average element temperature, which is the basis of material property selection for the element. If thermal loads are applied, node temperatures are used to establish the temperature field within the element, and the temperature interpolation functions are the same as those assumed to represent element displacements.

1. Control Card (10I5)

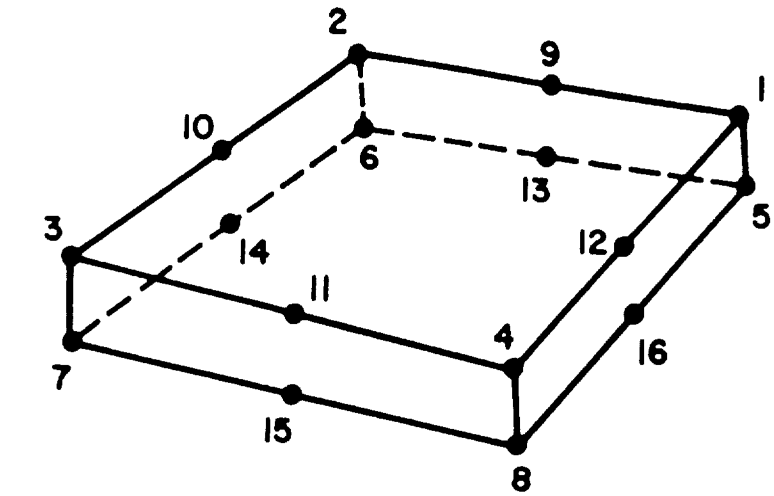
notes	columns	variable	entry
	5		Enter the number "8"
	6 - 10	NSOL21	Number of solid elements; GE.1
	11 - 15	NUMMAT	Number of different materials; GE.1
(1)	16 - 20	MAXTP	Maximum number of temperature points used in the table for any material; EQ.0; default set to "1"
(2)	21 - 25	NORTHO	Number of different sets of material axis orientation data; EQ.0; all properties are defined in the X,Y,Z, system
(3)	26 - 30	NDLS	Number of different distributed load (i.e., pressure) sets
(4)	31 - 35	MAXNOD	Maximum number of nodes used to describe any one element; GE.8 and LE.21 EQ.0; default set to "21"
(5)	36 - 40	NOPSET	Number of sets of data requesting stress output at various element locations; EQ.0; centroid output only



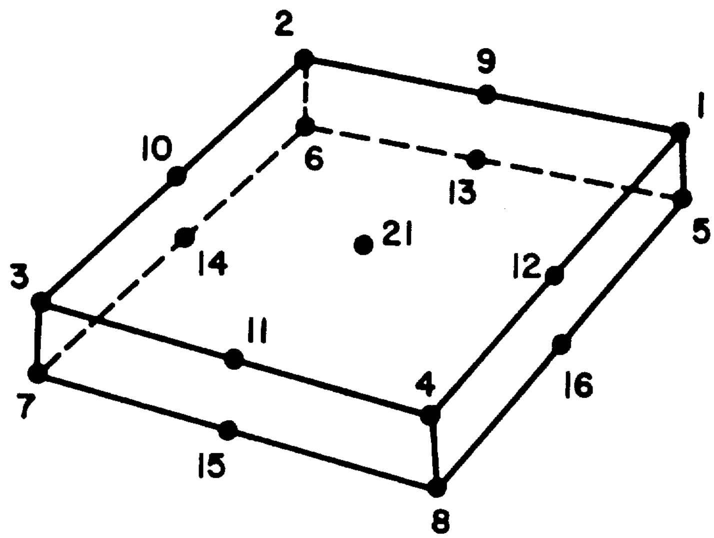
THREE DIMENSIONAL ISOPARAMETRIC ELEMENT



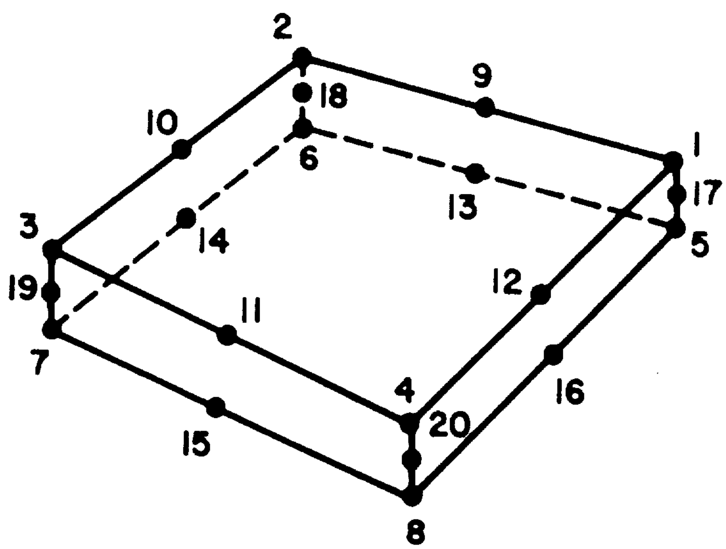
HEXAHEDRAL ELEMENT IN NATURAL COORDINATES



a. 16 – NODE ELEMENT



b. 17 – NODE ELEMENT



c. 20 – NODE ELEMENT

COMMONLY USED ELEMENT GEOMETRIES

IV. ELEMENT DATA (continued)

1. Control Card (10I5) (continued)

notes	columns	variable	entry
(6)	41 - 45	INTRS	Standard integration order for the natural (r,s) directions; GE.2 and LE.4 EQ.0; default set to "2"
	46 - 50	INTT	Standard integration order for the natural (t)-direction; GE.2 and LE.4 EQ.0; default set to "2"

NOTES/

(1) The variable MAXTP limits the number of temperature points that can be input for any one of the NUMMAT material sets; i.e., the variable NTP in Section 2 cannot exceed the value of MAXTP.

(2) NORTHO specifies the number of cards to be read in Section 3, and if omitted, all orthotropic material axes are assumed to coincide with the global cartesian axes X,Y,Z.

(3) NDLS specifies the number of card pairs to be read in Section 4. NDLS must be a positive integer if any pressure loads are to be applied to solid element faces.

(4) MAXNOD specifies the maximum number of non-zero node numbers assigned to any one of the NSOL21 elements input in Section 7. Locations of the element's 21 possible nodes are shown in the figure below in which the element is shown mapped into its natural r,s,t coordinate system. The eight corner nodes must be input for every element, and nodes 9 to 21 are input optionally. If MAXNOD is 9 or greater, all 21 node entries are read for each element (Cards 2 and 3, Section 7), but only the first MAXNOD non-zero entries encountered when reading in sequence from 1 to 21 will be used for element description. As an example, for the 16-17- and 20-node elements MAXNOD has values of 16, 17, 20, respectively.

(5) As a means of controlling the amount of solution output, stress output location sets are defined in Section 5, and the total number of these output requests is specified by the variable NOPSET. For the case of NOPSET.EQ.0, no data is input in Section 5, and the only stress output produced by the program is at the element centroid. Otherwise, stress output can be requested at up to seven (7) locations (selected from a table of 27 possible locations) by means of the data entries given in Section 5.

NOTES (continued)

(6) The entries INTRS and INTT control the number of integration points to be used in numerical evaluation of integrals over volumes in the (r,s) and (t)-coordinate directions, respectively. When solid elements are used to represent shell structures, the through-the-thickness integrations (i.e., in the natural t-axis direction) can be evaluated less accurately than those in-plane (i.e., in the r,s plane). For this case INTRS might be 3 and INTT would be chosen typically as 2. The entries INTRS and INTT are standard or reference values and are used if the integration order entries on the element cards (Card 1, Section 7) are omitted. Non-zero entries for integration order(s) given on the element cards over-ride the standard values posted on this card.

2. Material Property Cards

Orthotropic, temperature dependent material properties are allowed. For each different material that is requested on the Control Card, the following set of data must be supplied (i.e., NUMMAT sets total):

a. Material identification card (2I5,2F10.0,6A6)

notes	columns	variable	entry
(1)	1 - 5	M	Material identification number; GE.1 and LE.NUMMAT
	6 - 10	NTP	Number of different temperatures at which properties are given; LE.MAXTP EQ.0; default set to "1"
(2)	11 - 20	WTDEN	Weight density of the material used to computed static gravity loads
	21 - 30	MASSDN	Mass density of the material used to compute the mass matrix in a dynamic analysis; EQ.0; default set to "WTDEN/386.4"
	31 - 66		Material description used to label the output.

NOTES/

(1) Material numbers (M) must be input in ascending sequence beginning with "1" and ending with "NUMMAT"; omissions or repetitions are illegal.

(2) Weight density is used to compute static node forces due to applied gravity loads; mass density is used to calculate element mass matrices for use in connection with a dynamic analysis.

IV. ELEMENT DATA (continued)

b. Material cards (7F10.0,6F10.0)

NTP pairs of cards are input in order of algebraically increasing value of temperature.

First Card

notes	columns	variable	entry
(1)	1 - 10		Temperature, T_n
(2)	11 - 20		E_{11} at T_n
	21 - 30		E_{22} at T_n
	31 - 40		E_{33} at T_n
	41 - 50		ν_{12} at T_n
	51 - 60		ν_{13} at T_n
	61 - 70		ν_{23} at T_n

Second Card

notes	columns	variable	entry
	1 - 10		G_{12} at T_n
	11 - 20		G_{13} at T_n
	21 - 30		G_{23} at T_n
	31 - 40		α_1 at T_n
	41 - 50		α_2 at T_n
	51 - 60		α_3 at T_n

NOTES/

(1) The 12 entries following the temperature value T_n are physical properties known at T_n. When two or more temperature points describe a material, interpolation based on average element temperature is performed to establish a property set for the element. Hence, the range of temperature points for a material table must span the expected range of average element temperatures for all elements associated with the material.

(2) The 12 constants $(E_{11}, E_{22}, \ldots, \alpha_3)$ are defined with respect to a set of axes (X_1, X_2, X_3) which are the principal material directions for an orthotropic, elastic medium. The stress-strain relations with respect to the (X_1, X_2, X_3) system is written as follows :

$$\begin{bmatrix} \varepsilon_{11} \\ \varepsilon_{22} \\ \varepsilon_{33} \\ \gamma_{12} \\ \gamma_{23} \\ \gamma_{31} \end{bmatrix} = \begin{bmatrix} 1/E_{11} & -\nu_{12}/E_{22} & -\nu_{13}/E_{33} & 0 & 0 & 0 \\ -\nu_{21}/E_{11} & 1/E_{22} & -\nu_{23}/E_{33} & 0 & 0 & 0 \\ -\nu_{31}/E_{11} & -\nu_{32}/E_{22} & 1/E_{33} & 0 & 0 & 0 \\ 0 & 0 & 0 & 1/G_{12} & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/G_{23} & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/G_{13} \end{bmatrix} \begin{bmatrix} \sigma_{11} \\ \sigma_{22} \\ \sigma_{33} \\ \tau_{12} \\ \tau_{23} \\ \tau_{31} \end{bmatrix}$$

$$- \quad [\Delta T\alpha_1 \quad \Delta T\alpha_2 \quad \Delta T\alpha_3 \quad 0 \quad 0 \quad 0]^T$$

where ε_{ii} and σ_{ii} are normal strains and stresses in the X_i directions; γ_{ij} and τ_{ij} are shear strains and stresses on the principal material planes; α_i are the coefficients of thermal expansion, and ΔT is the increase in temperature from stress free distributed over the element volume.

3. Material Axes Orientation Sets (4I5)

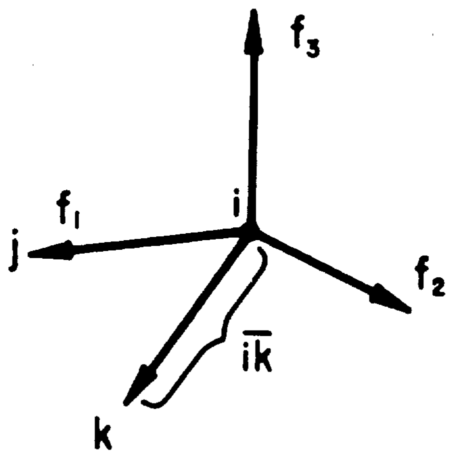
If NORTHO is zero on the Control Card, skip this data section, and all material axes (X_1,X_2,X_3) will be assumed to coincide with the global cartesian system X,Y,Z. Otherwise, NORTHO cards must be input as follows:

notes	columns	variable	entry
(1)	1 - 5	M	Identification number; GE.1 and LE.NORTHO
(2)	6 - 10	NI	Node number for point "i"
	11 - 15	NJ	Node number for point "j"
	16 - 20	NK	Node number for point "k"

NOTES/

(1) Identification numbers (M) must be input in increasing sequence beginning with "1" and ending with "NORTHO".

(2) Orthotropic material axes orientations are specified by means of the three node numbers NI,NJ,NK. For the special case where orthotropic material axes coincide with the global axes (X,Y,Z), it is not necessary to input data in this section; see Section 7, note (4). Let $\underline{f}_1$,$\underline{f}_2$,$\underline{f}_3$ be the three orthogonal vectors which define the axes of material orthotropy, then their directions are as shown below:



$$\underline{f}_1 = \overleftrightarrow{ij}$$

$$\underline{f}_3 = \overleftrightarrow{ij} \times \overleftrightarrow{ik}$$

$$\underline{f}_2 = \underline{f}_3 \times \underline{f}_1$$

Node numbers NI,NJ,NK are only used to locate points i,j,k, respectively, and any convenient nodes may be used.

4. Distributed Surface Load Data

NDLS pairs of cards are to be input in this section in order of increasing set number (N). These data describe surface loads acting on element faces and may be prescribed directly in terms of face corner node pressures or indirectly by means of a hydrostatic pressure field.

a. Control Card (3I5)

notes	columns	variable	entry
(1)	1 - 5	N	Load set identification number; GE.1 and LE.NDLS
(2)	6 - 10	NFACE	Element face number on which this distributed load is acting; GE.1 and LE.6
(3)	11 - 15	LT	Load type code; EQ.1; prescribed normal pressure intensities EQ.2; hydrostatically varying pressure field EQ.0; default set to "1"

IV. ELEMENT DATA (continued)

NOTES/

(1) The surface load data sets established in this section are assigned to the elements in Section 7.

(2) Hexahedra have six quadrilateral faces each uniquely described by four node numbers at the corners of the face. The face number convention established for elements is given in the Table below.

(3) Two types of surface pressure loads may be applied to faces of the elements. If LT.EQ.0 (or 1), a normal pressure distribution is prescribed directly by means of pressure intensities at the face corner nodes. If LT.EQ.2, the face is exposed to hydrostatic pressure due to fluid head.

FACE NUMBER	NATURAL COORDINATES	CORNER NODE NUMBERS N_1	N_2	N_3	N_4
1	(+1, s, t)	1	4	8	5
2	(-1, s, t)	2	3	7	6
3	(r,+1, t)	1	5	6	2
4	(r,-1, t)	4	8	7	3
5	(r, s,+1)	1	2	3	4
6	(r, s,-1)	5	6	7	8

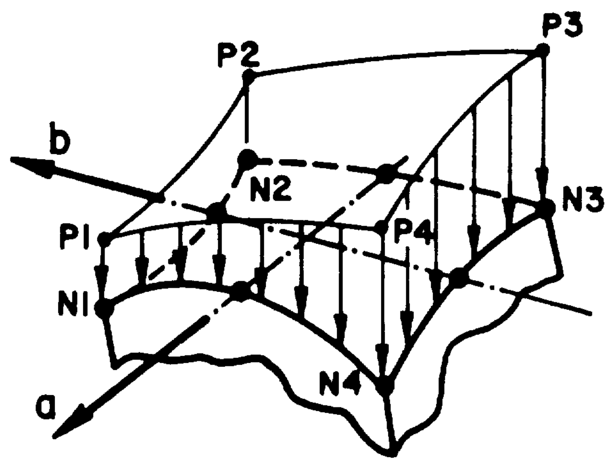
TABLE Corner Node Numbers for the Solid Element Faces

b. Normal Pressure Data (4F10.0) (LT.EQ.1, only)

notes	columns	variable	entry
(1)	1 - 10	P1	Pressure at face node N_1
(2)	11 - 20	P2	Pressure at face node N_2; EQ.0; default set to "P1"
	21 - 30	P3	Pressure at face node N_3; EQ.0; default set to "P1"
	31 - 40	P4	Pressure at face node N_4; EQ.0; default set to "P1"

NOTES/

(1) The pressure distribution acting on an element face is defined by specifying intensities P1,P2,P3,P4 at the face corner nodes as shown below:



The face corner node numbers are given in the Table and positive pressure tends to compress the volume of the element.

The variation of pressure over the element face, p(a,b), is given as:

$$p(a,b) = P1 \times h_1 + P2 \times h_2 + P3 \times h_3 + P4 \times h_4$$

where

$$h_1 = (1/4)\ (1+a)\ (1+b)$$
$$h_2 = (1/4)\ (1-a)\ (1+b)$$
$$h_3 = (1/4)\ (1-a)\ (1-b)$$
$$h_4 = (1/4)\ (1+a)\ (1-b)$$

in quadrilateral natural face coordinates (a,b).

(2) If any of the entries P2,P3,P4 are omitted, these values are re-set to the value of P1; i.e., for a uniformly distributed pressure (p), we have P1.EQ.p and cc 11-40 blank. If P2 is zero specify a small number.

IV. ELEMENT DATA (continued)

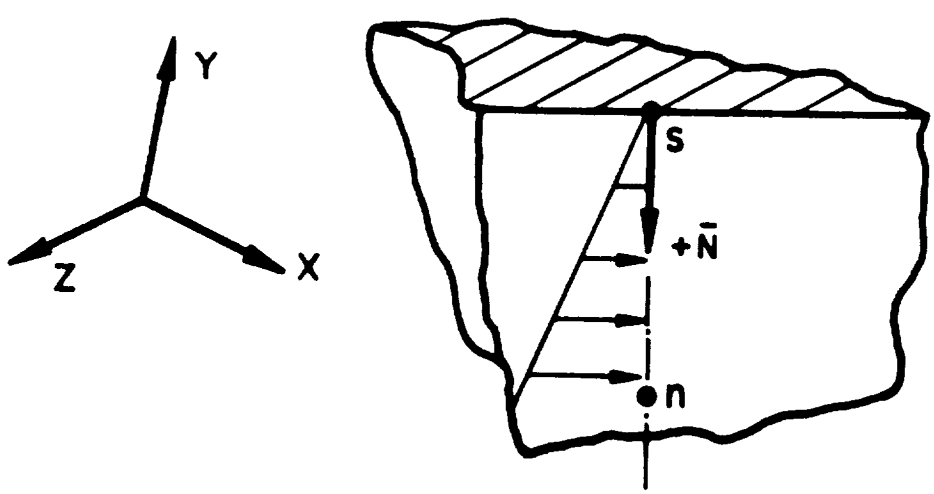
c. Hydrostatic Pressure Data (7F10.0) (LT.EQ.2, only)

notes	columns	variable	entry
(1)	1 - 10	GAMMA	Weight density of the fluid, γ; GT.0
(2)	11 - 20	XS	X-ordinate of point s in the free surface of the fluid
	21 - 30	YS	Y-ordinate of point s in the free surface of the fluid
	31 - 40	ZS	Z-ordinate of point s in the free surface of the fluid
	41 - 50	XN	X-ordinate of a point n on the normal to the fluid surface
	51 - 60	YN	Y-ordinate of a point n on the normal to the fluid surface
	61 - 70	ZN	Z-ordinate of a point n on the normal to the fluid surface

NOTES/

(1) GAMMA is the weight density (i.e., units of force per unit of fluid volume) of the fluid in contact with element face number NFACE.

(2) Point "s" is any point in the free surface of the fluid, and point "n" is located such that the direction from s to n is normal to the free surface and is positive with increasing depth.



Hydrostatic pressure in contact with an element face causes element compression; i.e., pressure resultant acts toward the element centroid. Nodes located above the fluid surface are automatically assigned zero pressure intensities if an element face is not (or only partially) submerged in the fluid.

5. Stress Output Request Location Sets (7I5)

If NOPSET is zero on the Control Card, skip this section, and global stresses will be computed and output at the element centroid only. Otherwise, NOPSET cards must be input as follows:

notes	column	variable	entry
(1)	1 - 5	LOC1	Location number of output point 1
	6 - 10	LOC2	Location number of output point 2
	11 - 15	LOC3	Location number of output point 3
	16 - 20	LOC4	Location number of output point 4
	21 - 25	LOC5	Location number of output point 5
	26 - 30	LOC6	Location number of output point 6
	31 - 35	LOC7	Location number of output point 7 LE. 27

NOTES/

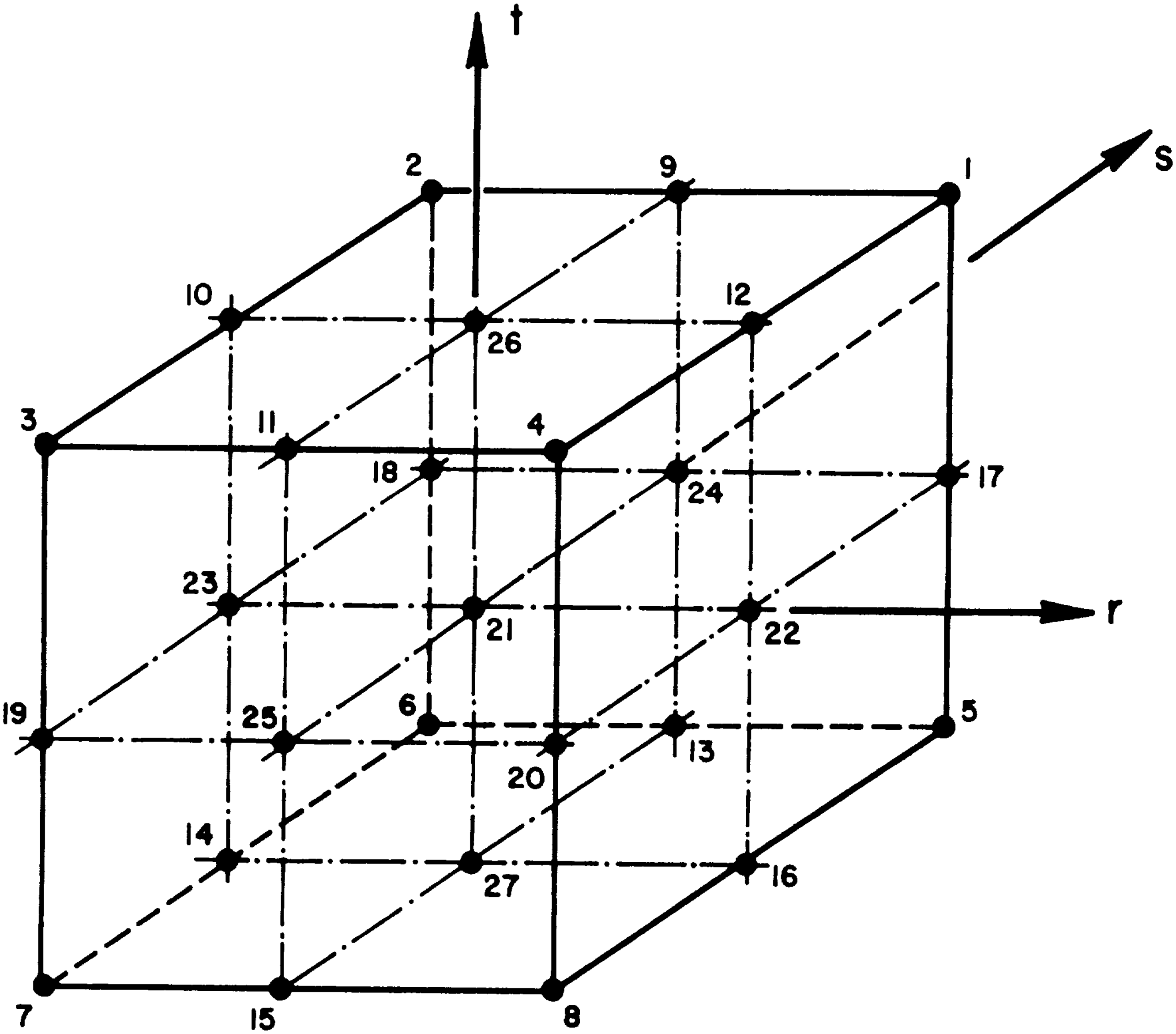
(1) 27 element locations are assigned numbers as shown in the Figure below. Locations 1 to 21 correspond to node numbers 1 to 21, respectively. Locations 22 to 27 are element face centroids. The first zero (or blank) entry on a location card terminates reading of location numbers for the output set; hence, fewer than seven locations can be requested in an output set. Location numbers must be input in order of increasing magnitude; i.e., LOC2 is greater than LOC1, LOC3 is greater than LOC2, etc. In dynamic analysis, FACE 1, FACE 2,..., FACE 6 correspond to output locations 22,23,...,27 respectively. (See Table VII.1).

6. Element Load Case Multipliers

Five (5) cards must be input in this section specifying the fraction of gravity (X,Y,Z), the fraction of thermal loads and the fraction of pressure loads to be added to each of the element loading combinations (A,B,...). Load case multiplier data affect static analysis calculations only.

Card 1 X-direction gravity (4F10.0)

notes	columns	variable	entry
(1)	1 - 10	GXA	Fraction of X-direction gravity to be applied in element load case A
			. . .
	31 - 40	GXD	Fraction of X-direction gravity to be applied in element load case D



ELEMENT STRESS OUTPUT LOCATION NUMBERS

IV. ELEMENT DATA (continued)

Card 2 Y-direction gravity (4F10.0)

Card 3 Z-direction gravity (4F10.0)

Card 4 Thermal loads (4F10.0)

notes	columns	variable	entry
(2)	1 - 10	TA	Fraction of thermal loads to be applied in element load case A
			. . .
	31 - 40	TD	Fraction of thermal loads to be applied in element load case D

Card 5 Pressure loads (4F10.0)

notes	columns	variable	entry
(3)	1 - 10	PA	Fraction of pressure loads to be applied in element load case A
			. . .
	31 - 40	PD	Fraction of pressure loads to be applied in element load case D

NOTES:

(1) Gravity loads on the structure due to static body forces are computed from the weight density of element materials and the element geometry. These loads are assigned to the element load combinations by means of the entries on Cards 1,2 and 3 for forces in the X,Y,Z directions, respectively.

(2) Thermal loads are computed knowing the node temperatures input in Section III, the stress free reference temperature (T_o) input in Section 7 and the element's material properties and node coordinates. The temperature distribution within the element is described using the same interpolation functions which describe the variation of displacements within the element.

(3) Pressure loads are first assigned to element load cases (A,B,...) by means of the entries (scale factors) on Card 5, and the distributed load sets which were input in Section 4 are then applied to the elements individually for cases (A,B,...) by means of load set references given in Section 7.

7. Element Cards

Two cards (if MAXNOD.EQ.8) or three cards (if MAXNOD.GT.8) must be prepared for each element that appears in the input, and the

IV. ELEMENT DATA (continued)

format for these cards is as follows:

Card 1 (6I5,F10.,4I5,4I2)

notes	columns	variable	entry
(1)	1 - 5	M	Element number; GE.1 and LE.NSOL21
(2)	6 - 10	NDIS	Number of nodes to be used in describing the element's displacement field; EQ.0; default set to "MAXNOD"
(3)	11 - 15	NXYZ	Number of nodes to be used in the description of element geometry; EQ.0; default set to "NDIS" EQ.NDIS → isoparametric element LT.NDIS → subparametric element
	16 - 20	NMAT	Material identification number; GE.1 and LE.NUMMAT
(4)	21 - 35	MAXES	Identification number of the material axis orientation set; GE.1 and LE.NORTHO EQ.0; material axes default to the global X,Y,Z system
(5)	26 - 30	IOP	Identification number of the stress output location set; GE.1 and LE.NOPSET EQ.0; centroid output only
	31 - 40	TZ	Stress free reference temperature, T_o
(6)	41 - 45	KG	Node number increment for element data generation; EQ.0; default set to "1"
	46 - 50	NRSINT	Integration order for natural coordinate (r,s) directions; EQ.0; default set to "INTRS"
	51 - 55	NTINT	Integration order for natural coordinate (t) direction; EQ.0; default set to "INTT"
(7)	56 - 60	IREUSE	Flag indicating that the stiffness and mass matrices for this element are the same as those for the preceding element; EQ.0; no EQ.1; yes
(8)	61 - 62	LSA	Pressure set for element load case A
	63 - 64	LSB	Pressure set for element load case B
	65 - 66	LSC	Pressure set for element load case C
	67 - 68	LSD	Pressure set for element load case D; LE.NDLS

IV. ELEMENT DATA (continued)

Card 2 (16I5)

notes	columns	variable	entry
(9)	1 - 5		Node 1 number
	6 - 10		Node 2 number
	11 - 15		Node 3 number
	16 - 20		Node 4 number
	21 - 25		Node 5 number
	26 - 30		Node 6 number
	31 - 35		Node 7 number
	36 - 40		Node 8 number
(10)	41 - 45		Node 9 number
	46 - 50		Node 10 number
	51 - 55		Node 11 number
	56 - 60		Node 12 number
	61 - 65		Node 13 number
	66 - 70		Node 14 number
	71 - 75		Node 15 number
	76 - 80		Node 16 number

Card 3 (5I5) (required if MAXNOD.GT.8)

note	columns	variable	entry
	1 - 5		Node 17 number
	6 - 10		Node 18 number
	11 - 15		Node 19 number
	16 - 20		Node 20 number
	21 - 25		Node 21 number

NOTES/

(1) Element cards must be input in ascending element number order beginning with "1" and ending with "NSOL21". Repetition of element numbers is illegal, but element cards may be omitted, and missing element data are generated according to the procedure described in note (7).

(2) NDIS is a count of the node numbers actually posted on Cards 2 and 3 which must immediately follow Card 1. NDIS must be at least eight (8), but must be less than or equal to the limit (MAXNOD) which was given on the Control Card, Section 1. Element displacements are assigned at the NDIS non-zero nodes, and thus, the order of the element matrices is three (i.e., translations X,Y,Z) times NDIS. The eight corner nodes of the hexahedron must be input, but nodes 9 to 21 are optional, and any or all of these optional nodes may be used to describe the element's displacement field.

(3) When element edges are straight it is unnecessary computationally to include side nodes in the numerical evaluation of coordinate derivatives, the Jacobian matrix, etc., and since regular element shapes are common, an option has been included to use fewer nodes in these geometric calculations than are used to describe element displacements. The first NXYZ non-zero nodes posted on Cards 2 and 3 are used to evaluate those parameters which pertain to element geometry only. NXYZ must be at least eight (8), and if omitted is re-set to NDIS. A common application might be a 20 node element (i.e., NDIS.EQ.20) with straight edges in which case NXYZ would be entered as "8".

(4) MAXES (unless omitted) refers to one of the material axes set defined in Section 3. If omitted, the material (NMAT) orientation is such that the (X_1, X_2, X_3) axes coincide with the (X,Y,Z) axes, respectively.

(5) IOP (unless omitted) refers to one of the output location sets given in Section 5. If IOP.EQ.0, stress output is quoted at the element centroid only. Stress output at a point consists of three normal and three shear components referenced to the global (X,Y,Z) axes.

(6) When element cards are omitted, element data are generated automatically as follows:

   (a) all data on Card 1 for generated elements is taken to be the same as that given on the first element card in the sequence;

   (b) non-zero node numbers (given on Cards 2 and 3 for the first element) are incremented by the value "KG" (which is given on Card 1 of the first element) as element generation progresses; zero (or blank) node number entries are generated as zeroes.

The last element cannot be generated.

(7) The flag IREUSE allows the program to bypass stiffness and mass matrix calculations providing the current element is identical to the preceding element; i.e., the preceding and current elements are identical except for a rigid body translation. If IREUSE.EQ.0, new matrices are computed for the current element. If IREUSE.EQ.1 it is also assumed that the node temperatures of the element (for calculation of thermal loads) are the same as those of the preceding element.

IV. ELEMENT DATA (continued)

(8) Pressure loads are assigned (i.e., applied) to the element by means of load set references in cc 61-62 for combination A, cc 63-64 for B, etc. A zero entry means that no pressure acts on the element for that particular element load combination.

(9) The first eight node numbers establish the corners or vertices of a general hexahedron and must be all non-zero, (see Figure in Section 1 on control cards). Node numbers must be input in the sequence indicated otherwise volume and surface area integrations will be indefinite.

(10) The number of cards required as input for each element depends on the variable MAXNOD. For the case of MAXNOD.EQ.8, only Card 2 is required. If MAXNOD.GT.8, Cards 2 and 3 are required for all elements.

Nodes 9 to 21 are optional, and only those nodes actually used to describe the element are input. The program will read all 21 entries if MAXNOD was given as 9 or greater, but only NDIS non-zero values are expected to be read on Cards 2 and 3. If for example one element is described by 10 nodes, then cc 1-40 on Card 2 would be the eight corner node numbers, and the remaining two node numbers would be posted somewhere on Cards 2 and 3.

IV. ELEMENT DATA (continued)

TYPE 9 - THREE-DIMENSIONAL STRAIGHT OR CURVED PIPE ELEMENTS

Pipe elements are identified by the number twelve (12). Axial and shear forces, torque and bending moments are calculated for each member. Gravity loadings in the global (X,Y,Z) directions, uniform temperature changes (computed from input nodal temperatures), and extensional effects due to internal pressure form the basic member loading conditions. Pipe element input is described by the following sequence of cards:

1. Control Card (14I5)

notes	columns	variable	entry
	4 - 5		Enter the number "12"
(1)	6 - 10	NPIPE	Number of pipe elements
	11 - 15	NUMMAT	Number of material sets
	16 - 20	MAXTP	Maximum number of temperature points used in the table for any material GE.1; at least one point
	21 - 25	NSECT	Number of section property sets; GE.1
(2)	26 - 30	NBRP	Number of branch point nodes at which output is required; EQ.0; no branch point output is produced
	31 - 35	MAXTAN	Maximum number of tangent elements common to any one branch point node; EQ.0; default set to "4"
	36 - 40	NPAR(8)	Blank
	41 - 45	NPAR(9)	Tangent stiffness load matrix dump flag EQ.1; Print EQ.0; Suppress printing
	46 - 50	NPAR(10)	Bend stiffness load matrix dump flag EQ.1; Print EQ.0; Suppress printing
	51 - 55	NPAR(11)	Element parameters dump flag EQ.1; Print EQ.0; Suppress printing

NOTES/

(1) The number of pipe elements ("NPIPE") counts both tangent and bend geometries, and both the material and section property tables can reference either the bend or tangent element types.

(2) A branch point is defined as a nodal location where at least three (3) tangent pipe elements connect. The two input parameters "NBRP" and "MAXTAN" reserve storage for an index array created during the processing of pipe element data; posting a larger number of maximum common tangents than actually exist is not considered a fatal error condition. Branch point data is read if requested, but not currently used; i.e. to be used in future program versions.

IV. ELEMENT DATA (continued)

2. Material Property Cards

Temperature-dependent Young's modulus (E), Poisson's ratio (ν) and thermal expansion coefficient (α) are allowed. If more than one (1) temperature point is input for a material table, then the program selects properties using linear interpolation between input temperature values. The temperature used for property selection is the average element temperature which is denoted as T_a:

$$T_a = (T_i + T_j)/2$$

where T_i and T_j are the input nodal temperatures for ends "i" and "j" of the pipe. For each different material, the following set of cards must be input:

a. material identification card (2I5,6A6)

notes	columns	variable	entry
(1)	1 - 5	M	Material identification number; GE.1 and LE.NUMMAT
	6 - 10	NT	Number of different temperatures at which properties are given; EQ.0; one temperature point is assumed to be input
	11 - 46		Material description used to label the output for this material

NOTES/

(1) Material identification number must be input between one ("1") and the total number of materials specified ("NUMMAT")

b. material cards (4F10.0)

notes	columns	variable	entry
(1)	1 - 10	T(N)	Temperature, T_n
	11 - 20	E(N)	Young's modulus, E_n
	21 - 30	XNU(N)	Poisson's ratio, ν_n
	31 - 40	ALP(N)	Thermal expansion coefficient, α_n

NOTES/

(1) Supply one card for each temperature point in the material table; at least one card is required. Temperatures must be input in increasing (algebraic) order. If two or more points are used, care must be taken to insure that the table covers the expected range of average temperatures existing in the elements to which the material table is assigned.

3. Section Property Cards (I5,5F10.0,3A6)

notes	columns	variable	entry
(1)	1 - 5	N	Section property identification number; GE.1 and LE.NSECT
(2)	6 - 15		Outside diameter of the pipe, d_o
	16 - 25		Pipe wall thickness, t
	26 - 35		Shape factor for shear distortion, α_v
(3)	36 - 45		Weight per unit length of section, γ_1
(4)	46 - 55		Mass per unit length of section, ρ_1
	56 - 73		Section description (used to label the output)

NOTES/

(1) Section property identification numbers must be input in an ascending sequence beginning with one ("1") and ending with the total number of section specified ("NSECT").

(2) Assuming that (y,z) are the section axes and that the x-axis is normal to the section, the properties for the section are computed from the input parameters [d_o, t and α_v] as follows:

(a) inner and outer pipe radii;

$$r_o = d_o/2$$

$$r_i = r_o - t$$

(b) cross sectional area (axial deformations);

$$A_x = \pi(r_o^2 - r_i^2)$$

(c) principal moments of inertia (bending);

$$I_y = (\pi/4)\ (r_o^4 - r_i^4)$$

$$I_z = I_y$$

(d) polar moment of inertia (torsion);

$$J_x = 2I_y$$

(e) effective shear areas (shear distortions);

$$A_y = A_x/\alpha_v$$

$$A_z = A_y$$

Note that the shape factor for shear distortion (α_v) may be input directly. If the entry is omitted, the shape factor is computed using the equation:

$$\alpha_v = (4/3)\ (r_o^3 - r_i^3)/[(r_o^2 + r_i^2)\ (r_o - r_i)]$$

$$\rightarrow 2.0$$

IV. ELEMENT DATA (continued)

An input value for α_v greater than one hundred (100.) causes the program to neglect shear distortions entirely. If used, the same shape factor is applied to both in and out-of-plane shear distortions.

(3) The weight per unit length of section (γ_1) is used to compute gravity loadings on the elements. Fixed end shears, moments, torques, etc. are computed automatically and applied as equivalent nodal loads. These forces will not act on the structure unless first assigned to one of the element load cases (A,B,C,D) in Section IV.L.5, below.

(4) The mass per unit length is only used to form the lumped mass matrix for a dynamic analysis case. If no entry is input, then the program will re-define the mass density from the weight density using:

$$\rho_1 = \gamma_1/386.4$$

Either a non-zero weight density or mass density will cause the program to assign masses to all pipe element nodes.

4. Branch Point Node Numbers

If the number of output branch point nodes has been omitted from the control card (i.e., cc 26-30 blank), skip this section of input, and no branch point data will be read. Otherwise, supply node numbers for a total number of branch points requested on the control card, ten (10) nodes per card:

first card (10I5)

notes	columns	variable	entry
(1)	1 - 5		Node number at branch point 1
	6 - 10		Node number at branch point 2
	. . .		. . .
	45 - 50		Node number at branch point 10

second card (10I5) -- if required

notes	columns	variable	entry
	1 - 5		Node number at branch point 11
	. . .		. . .

NOTES.

(1) A node does not define a branch point unless at least three (3) tangent elements are common to the node. Branch point output is only produced for static analysis cases.

IV. ELEMENT DATA (continued)

5. Element Load Case Multipliers

Five (5) cards must be input in this section specifying the fraction of gravity (in each of the X,Y,Z coordinate directions), the fraction of thermal loading and the fraction of internal pipe pressure loading to be added to each of four (4) possible element loading combinations (A,B,C,D).

Card 1 X-direction gravity (4F10.0)

notes	columns	variable	entry
(1)	1 - 10		Fraction of X-direction gravity to be applied in element load case A
	11 - 20		Fraction of X-direction gravity to be applied in element load case B
	21 - 30		Fraction of X-direction gravity to be applied in element load case C
	31 - 40		Fraction of X-direction gravity to be applied in element load case D

Card 2 Y-direction gravity (4F10.0)

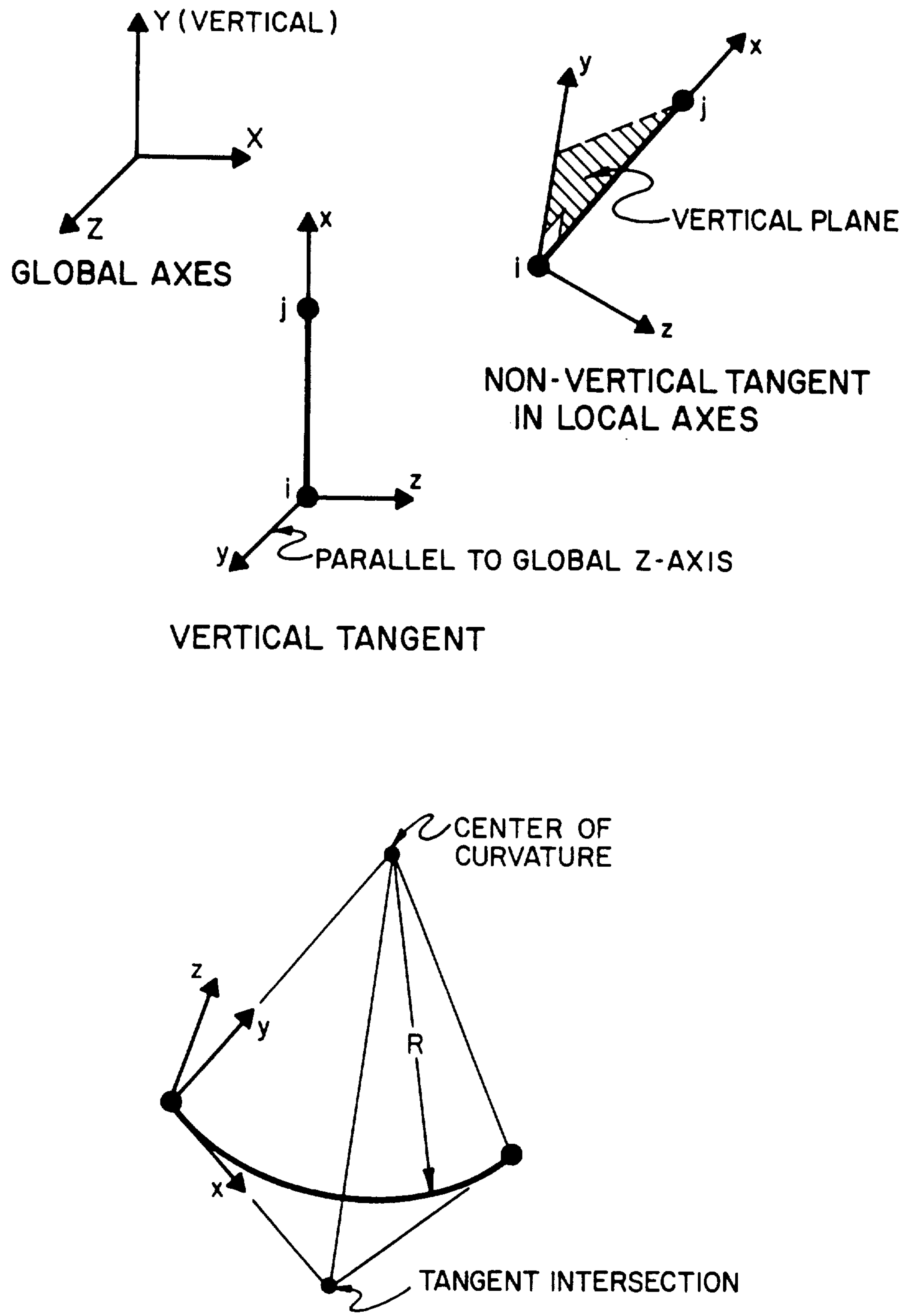
Card 3 Z-direction gravity (4F10.0)

Card 4 Thermal loads (4F10.0)

notes	columns	variable	entry
(2)	1 - 10		Fraction of thermal loading to be applied in element load case A
	11 - 20		Fraction of thermal loading to be applied in element load case B
	21 - 30		Fraction of thermal loading to be applied in element load case C
	31 - 40		Fraction of thermal loading to be applied in element load case D

Card 5 Internal pressure (4F10.0)

notes	columns	variable	entry
(3)	1 - 10		Fraction of pressure-induced loading applied in element load case A
	11 - 20		Fraction of pressure-induced loading applied in element load case B
	21 - 30		Fraction of pressure-induced loading applied in element load case C
	31 - 40		Fraction of pressure-induced loading applied in element load case D



LOCAL COORDINATE SYSTEMS FOR
PIPE ELEMENTS

IV. ELEMENT DATA (continued)

5. Element Load Case Multipliers (continued)

NOTES.

(1) No gravity loads will be produced if the weight per unit length was input as zero on all section property cards. Otherwise, a multiplier of 1.0 input for an element load case means that 100% of deadweight will be assigned to that load combination.

(2) No thermal loading will result if the coefficient of thermal expansion has been omitted from all the material cards. Otherwise, thermal loads are computed for each element using the ΔT between the average element temperature (T_a) and the stress-free temperature (T_o) given with each pipe element card (Section IV.L.6, below).

(3) Element distortions are computed for each element due to internal pressure, and these loads are combined into element load cases by means of appropriate non-zero entries in Card 5.

Gravity, thermal or pressure induced loads cannot act on the structure unless first combined in one or more of the element load sets (A,B,C,D). Once defined, element load cases are assigned (via scale factors) to the structure load cases by means of Element Load Multipliers given in Section VI. An element load case combination may be used a multiple number of times when defining the various structure loading conditions.

6. Pipe Element Cards

a. card type 1

notes	columns	variable	entry
(1)	1 - 4	N	Pipe element number; GE.1 and LE.NPIPE
	5		Geometric type code: "T" (or blank); tangent section "B" ; bend (circular) section
	6 - 10	I	Node I number
	11 - 15	J	Node J number
	16 - 20	MAT	Material identification number; GE.1 and LE.NUMMAT
	21 - 25	ISECT	Section property identification number; GE.1 and LE.NSECT
(2)	26 - 35		Stress-free temperature, T_o
(3)	36 - 45		Internal pressure, p
(4)	46 - 55		Positive projection of a local y-vector on the global X-axis; A(yX)

IV. ELEMENT DATA (continued)

6. Pipe Element Cards (continued)

notes	columns	variable	entry
	56 - 65		Positive projection of a local y-vector on the global Y-axis; A(yY)
	66 - 75		Positive projection of a local y-vector on the global Z-axis A (yZ)
(5)	76 - 80	KG	Node number increment for tangent element generation; EQ.0; default set to "1"

NOTES/

(1) Card type 1 is used for both tangent and bend elements; a second card (card type 2, below) must be input immediately following card type 1 if the pipe element is a bend (i.e., "B" in cc 5). Note that element cards must be input in ascending sequence beginning with one ("1") and ending with the total number of pipe elements. If tangent elements are omitted, generation of the intermediate elements will occur; the generation algorithm is described below. An attempt to generate bend type elements is considered to be an error.

(2) The stress-free temperature, T_o, is subtracted from the average element temperature, T_a, to compute the uniform temperature difference acting on the element:

$$\Delta T = T_a - T_o$$

The entire element is assumed to be at this uniform value of temperature difference.

(3) The value of pressure is used to compute a set of self-equilibrating joint forces arising from member distortions due to pressurization; i.e., the mechanical equivalent of thermal loads. For bend elements, the pressure is also used to compute the bend flexibility factor, k_p. The curved pipe subjected to bending is more flexible than elementary beam theory would predict. The ratio of "actual" flexibility to that predicted by beam theory is denoted by k_p, where

$$k_p = (1.65/h)/[1 + (6p/Eh)(R/t)^{4/3}] \geq 1$$

in which

$$h = tR/r^2$$

$$r = (d_o - t)/2$$

6. Pipe Element Cards (continued)

and

t = pipe wall thickness
R = radius of the circular bend
r = mean radius of the pipe cross section
d_o = outside diameter of the pipe
E = Young's modulus
p = internal pressure

The flexibility factor is computed and applied to all bend elements; pressure stiffening is neglected if the entry for internal pressure ("p") is omitted.

(4) The global projections of the local y-axis for a tangent member may be omitted (cc 46-75 blank); for this case, the following convention for the local system is assumed:

(a) tangents parallel to the global Y-axis (vertical axis) have their local y-axes directed parallel to and in the same direction as the global Z-axis;

(b) tangents not parallel to the global Y-axis have their local y-axes contained in a vertical (global) plane such that local y projects positively on the positive global Y-axis.

For bend elements, the global projections of the local y-axis are not used; instead, the local axis convention is defined as follows:

(a) the local y-axis is directed positively toward and intersects the center of curvature of the bend (i.e., radius vector);

(b) the local x-axis is tangent to the arc of the bend and is directed positively from node I to node J.

Note that for all elements, the local x, y, z system is a right-handed set (see figure).

(5) If a tangent element sequence exists such that each element number (NE_i) is one (1) greater than the previous number (NE_{i-1}); i.e.,

$$NE_i = NE_{i-1} + 1$$

only the element card for the first tangent in the

6. Pipe Element Cards (continued)

series need be input. The node numbers for the missing tangents are computed using the formulae:

$$NI_i = NI_{i-1} + KG$$

$$NJ_i = NJ_{i-1} + KG$$

where "KG" is the node number increment input in cc 76-80 for the first element in the series, and the

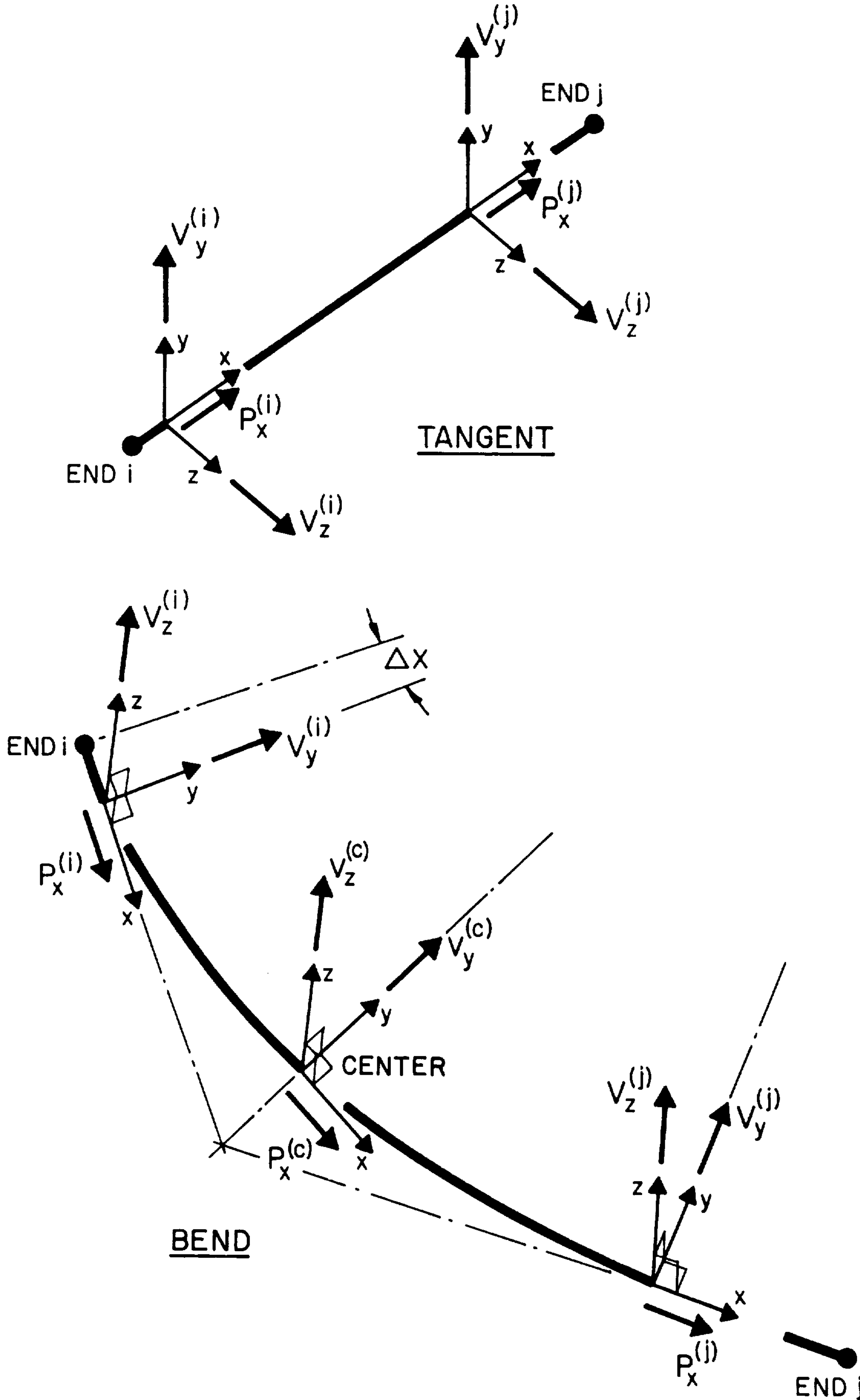
(a) material identification number
(b) section property identification number
(c) stress-free temperature
(d) internal pressure
(e) y-axis global projections

for each tangent in the generation sequence are taken to be the same as those input on the first card in the series. The node number increment ("KG") is reset to one (1) if left blank on the first card in the series. The last (highest) element cannot be generated; i.e., it must be input.

Bend element data cannot be generated because two input cards are required for each bend. Also, the element just prior to a bend element must appear on an input card. Several bends may be input in a sequence, but each bend must appear (on two cards) in the input stream.

b. card type 2 (F10.0,3X,A2,4F10.0)

notes	columns	variable	entry
(1)	1 - 10	R	Radius of the bend element, R
(2)	14 - 15		Third point type code: "TI" (or blank); third point is the tangent intersection point "CC" ; third point is the center of curvature
	16 - 25		X-ordinate of the third point, X_3
	26 - 35		Y-ordinate of the third point, Y_3
	36 - 45		Z-ordinate of the third point, Z_3
	46 - 55		Fraction of wall thickness to be used for dimensional tolerance tests; EQ.0; default set to "0.1"



FORCE SIGN CONVENTION FOR PIPE ELEMENT OUTPUT

IV. ELEMENT DATA (continued)

6. Pipe Element Cards (continued)

NOTES/

(1) The radius of the bend ("R") must be input regardless of the method ("TI" or "CC") used to define the third point for the bend.

(2) If the tangent intersection point is used, the program computes a radius for the bend and compares the computed value with the input radius. An error condition is declared if the two radii are different by more than the specified fraction (or multiple) of the section wall thickness. The lengths of the two tangent lines (I to TI and J to TI) are compared for equality, and an error will be flagged if the two values are discrepant by more than the dimensional tolerance.

If the center of curvature is input, the distances from the third point to nodes I and J are compared to the input radius; discrepancies larger than the user defined tolerance are noted as errors.

This second element card is only to be input for the bend type element.

Element Stress Output

Stress output for pipe elements consists of forces and moments acting in the member cross sections at the ends of each member and at the midpoints of the arcs in bend elements. Output quantitites act on the element segment connecting the particular output station and end i; i.e., j to i, center to i, or ΔX to i (where $\Delta X \to 0$). Positive force/moment vectors are directed into the positive local (x,y,z) directions, as shown in the accompanying figure.

V. CONCENTRATED LOAD/MASS DATA (2I5,6F10.4)

notes	columns	variable	entry
(1)	1 - 5	N	Nodal point number
(2)	6 - 10	L	Structure load case number; GE.1; static analysis EQ.0; dynamic analysis
	11 - 20	FX(N,L)	X-direction force (or translational mass coefficient)
	21 - 30	FY(N,L)	Y-direction force (or translational mass coefficient)
	31 - 40	FZ(N,L)	Z-direction force (or translational mass coefficient)
	41 - 50	MX(N,L)	X-axis moment (or rotational inertia)
	51 - 60	MY(N,L)	Y-axis moment (or rotational inertia)
	61 - 70	MZ(N,L)	Z-axis moment (or rotational inertia)

NOTES/

(1) For a static analysis case (NDYN.EQ.0), one card is required for each nodal point ("N") having applied (non-zero) concentrated forces or moments. All structure load cases must be grouped together for the node ("N") before data is entered for the next (higher) node at which loads are applied. Only the structure load cases for which node N is loaded need be given, but the structure load case numbers ("L") which are referenced must be supplied in ascending order. Node loadings must be defined (input) in increasing node number order, but again, only those nodes actually loaded are required as input. The static loads defined in this section act on the structure exactly as input and are not scaled, factored, etc. by the element load case (A,B,C,D) multipliers (Section VI, below). Nodal forces arising from element loadings are combined (additively) with any concentrated loads given in this section. Applied force/moment vectors act on the structure, positive in the positive global directions. Only one card is allowed per node per load case.

For a dynamic analysis case (NDYN.EQ.1,2, 3 or 4), structure load cases have no meaning, but the program expects to read data in this section nonetheless. In place of concentrated loads, lumped mass coefficients for the nodal degrees of freedom may be input for any (or all) nodes. The mass matrix is automatically constructed by the program from element geometry and associated material densities; the mass coefficients read in this section are combined (additively) with the existing element-based lumped mass matrix. For mass input, a node may only be specified once, and the load case number ("L") must be zero (or blank).

V. CONCENTRATED LOAD/MASS DATA (2I5,6F10.4) (continued)

The program terminates reading loads (or mass) data when a zero (or blank) node number ("N") is encountered; i.e., terminate this section of input with a blank card. For the special case of a static analysis with no concentrated loads applied, input only one (1) blank card in this section. Similarly, a dynamic analysis in which the mass matrix is not to be augmented by any entries in this section requires only one (1) blank card as input.

(2) For a static analysis, structure load case numbers range from "1" to the total number of load cases requested on the Master Control Card ("LL"); thus, $1 \leq L \leq LL$, NDYN.EQ.0. For a dynamic analysis, only zero (0) references are allowed; thus, L = 0, NDYN.EQ.1,2 3, or 4.

V-a Optimal Design Data

VI. ELEMENT LOAD MULTIPLIERS (4F10.0) ADDCTF

notes	columns	variable	entry
(1,2)	1 - 10	EM(1)	Multiplier for element load case A
	11 - 20	EM(2)	Multiplier for element load case B
	21 - 30	EM(3)	Multiplier for element load case C
	31 - 40	EM(4)	Multiplier for element load case D

NOTES:

(1) One card must be given for each static (NDYN.EQ.0) structure load case requested on the Master Control Card ("LL"). The cards must reference load case numbers in ascending order. The four (4) element load sets (A,B,C,D), if created during the processing of element data (Section IV, above), are combined with any concentrated loads specified in Section V for the structure load cases. For example, suppose an analysis case calls for seven (7) static structure loading conditions (i.e., LL = 7), then the program expects to read seven (7) cards in this section. Further, suppose card number three (3) in this section contains the entries:

[EM(1),EM(2),EM(3),EM(4)] = [-3.0,0.0,2.0,0.0]

Structure load case three (3) will then be constructed using 100% of any concentrated loads specified in Section V minus (-) 300% of the loads in element set A plus (+) 200% of the loads in element set C. Load sets B and D will not be applied in structure load case 3. Element load sets may be referenced any number of times in order to construct different structure loading conditions. Element-based loads (gravity, thermal, etc.) can only be applied to the structure by means of the data entries in this section.

(2) If this case calls for one of the dynamic analysis options, supply only one blank card in this section. If the job is a dynamic re-start case (NDYN.EQ.-2 or -3), skip this section.

Static analysis input is complete with this section. Begin a new data case with a new Heading Card (see Section I).

VII. DYNAMIC ANALYSES

Four (4) types of dynamic analysis can be performed by the program. The type of analysis is indicated by the number "NDYN" specified in card columns 21-25 of the Master Control Card (Section II). If

NDYN.EQ.1; Determination of system mode shapes and frequencies only (complete input Section VII.A, only)

NDYN.EQ.2; Dynamic Response Analysis for arbitrary time dependent loads using mode superposition (complete both Sections VII.A and B below)

NDYN.EQ.3; Response Spectrum Analysis (complete both Sections VII.A and C, below)

NDYN.EQ.4; Dynamic Response Analysis for arbitrary time dependent loads using step-by-step direct integration (complete Section VII.B below)

In any given dynamic analysis case only one (1) value of NDYN will be considered. However, if NDYN.EQ.2 or 3, the program must first solve the eigenvalue problem for structure modes and frequencies. These eigenvalues/vectors are then used as input to either the Forced Response Analysis (NDYN.EQ.2) or to the Response Spectrum Analysis (NDYN.EQ.3). Hence, options 1, 2 or 3 all require that the control parameters for eigenvalue extraction be supplied in Section VII.A, below.

In case of a direct step-by-step integration analysis (NDYN.EQ.4) do not provide the eigenvalue solution control card of Section VII.A.

For the special case of dynamic analysis re-start (NDYN.EQ.-2 or -3), data input consists of the Heading Card (Section I), the Master Control Card (Section II), and either of Sections VII.B (-2) or VII.C (-3), below. Re-starting is possible only if a previous solution using the same model was performed with NDYN.EQ.1, and the results from this eigenvalue solution were saved on the re-start file. (See Appendix A.)

Up to this section the program processes (i.e., expects to read) essentially the same blocks of data for either the static or dynamic analysis cases; certain of these preceding data cards, however, are read by the program but are not used in the dynamic analysis phase. In general, the purpose of the preceding data sections is to provide information leading to the formation of the system stiffness and mass matrices (appropriately modified for displacement boundary conditions). For example, element load sets (A,B,C,D) may be constructed as though a static case were to be considered, but these data are not used in a dynamic analysis; i.e., the same data deck through Section IV can be used for either type of analysis. The concept of structure loading conditions is not defined for the dynamic case, and input for Sections V and VI must be prepared specially.

A diagonal (lumped) mass matrix is formed automatically using element geometry and assigned material density or densities. The mass matrix so defined contains only translational mass coefficients calculated from tributary element volumes common to each node. Known rotational inertias must be input for the individual nodal degrees of freedom in Section V, above.

Non-zero impressed displacements (or rotations) input by means of the BOUNDARY element (type "7") are ignored; instead the component is restrained against motion during dynamic motion of the structure.

The program does not change the order of the system by performing a condensation of those nodal degrees of freedom having no (zero) mass coefficients; i.e., a zero mass reduction is not performed. No distinction is made between static and dynamic degrees of freedom; i.e., they are identical in sequence, type and total number.

VII. DYNAMIC ANALYSES (continued)

A. MODE SHAPES AND FREQUENCIES (NDYN.EQ.1, 2 or 3) (3I5,2F10.0)

notes	columns	variable	entry
(1)	1 - 5	IFPR	Flag for printing intermediate matrices, norms, etc. calculated during the eigenvalue solution; EQ.0; do not print EQ.1; print
(2)	6 - 10	IFSS	Flag for performing the STURM SEQUENCE check; EQ.0; check to see if eigenvalues were missed EQ.1; pass on the check
(3)	11 - 15	NITEM	Maximum number of iterations allowed to reach the convergence tolerance; EQ.0; default set to "16"
(4)	16 - 25	RTOL	Convergence tolerance (accuracy) for the highest ("NF") requested eigenvalue; EQ.0; default set to "1.0E-5"
(5)	26 - 35	COFQ	Cut-off frequency (cycles/unit time) EQ.0; NF eigenvalues will be extracted GT.0; extract only those values below COFQ
(6)	36 - 40	NFO	Number of starting iteration vectors to be read from TAPE10
(7)	41 - 50	RF	What input should I use if I do not want to use a start option ?? [illegible]

NOTES/

(1) Extra output produced by the eigenvalue solutions can be requested; output produced by this option can be quite voluminous. Normal output produced by the program consists of an ordered list of eigenvalues followed by the eigenvectors for each mode. The number of modes found and printed is specified by the variable "NF" given in card columns 16-20 of the Master Control Card.

(2) The program performs the solution for eigenvalues/vectors using either of two (2) distinct algorithms:

(a) the DETERMINANT SEARCH algorithm requires that the upper triangular band of the system stiffness matrix fit into high speed memory (core); i.e., one equation "block".

(b) the SUBSPACE ITERATION algorithm is used if only portions (fractions) of the system matrix can be retained in core; i.e., the matrix (even though in band form) must be manipulated in blocks.

A. MODE SHAPES AND FREQUENCIES (continued)

The program will automatically select the SUBSPACE ITERATION procedure for eigenvalue solution if the model is too large for the in-core algorithm.

The entries "IFSS", "NITEM" and "RTOL" are ignored if the program can use the DETERMINANT SEARCH to find eigenvalues. Whether or not a model is too large for the DETERMINANT SEARCH depends on the amount of core allocated (by the programmer and not the user) for array storage. The program variable "MTOT" equals the amount of working storage available.

Define:

MBAND = maximum equation bandwidth (coefficients)
≃ (maximum element node number difference) X (average number of degrees of freedom per node)

NEQ = total number of degrees of freedom in the model
= (6) X (total number of nodes) - [number of fixed (deleted) degrees of freedom]

NEQB = number of equations per block of storage
≃ MTOT/ MBAND/ 2 (for large systems)

If NEQB is less than NEQ, the model is too large for the DETERMINANT SEARCH algorithm, and the SUBSPACE ITERATION procedure will be used.

If the SUBSPACE ITERATION algorithm is used the user may request that the STURM SEQUENCE check be performed. By experience the algorithm has always produced the lowest NF eigenvalues, but there is no formal mathematical proof that the calculated NF eigenvalues will always be the lowest ones. The STURM SEQUENCE check can be used to verify that the lowest NF eigenvalues have been obtained. It should be noted that the computational effort expended in performing the STURM SEQUENCE check is not trivial. A factorization of the complete system matrix is performed at a shift just to the right of the NFth eigenvalue.

If during the SUBSPACE ITERATION the NFth eigenvalue fails to converge to a tolerance of "RTOL" (normally 1.0E-5, or 5 significant figures) within "NITEM" (normally "16") iterations, then the STURM SEQUENCE flag ("IFSS") is ignored.

(3) The maximum number of iterations to reach convergence ("NITEM") applies only to the SUBSPACE ITERATION algorithm. If cc 11-15 are left blank, a default value of "16" for NITEM is assumed.

(4) The convergence tolerance ("RTOL") is applicable only if the SUBSPACE ITERATION algorithm is used. This tolerance test applies to the NFth eigenvalue, and all eigenvalues lower than the NFth one will be more accurate than RTOL. The lowest mode is found most accurately with precision decreasing with increasing mode number until the highest requested mode ("NF") is accurate to a tolerance of RTOL. Iteration is terminated after cycle number (k+1) if the NFth eigenvalue (λ, say) satisfies the inequality:

$$[\,|\lambda(k+1) - \lambda(k)|/\lambda(k)] < \text{RTOL}$$

If the determinant search algorithm is used, the eigenpairs are obtained to a high precision, which is indicated by the "physical error bounds"

$$\varepsilon_i = \|r_i\|_2 \,/\, \|K\phi_i\|_2$$

where

$$r_i = (K - \omega_i^2 M)\,\phi_i\ ,$$

and (ω_i^2 ϕ_i) are the i'th eigenvalue and eigenvector obtained in the solution.

(5) The cut-off frequency ("COFQ") is used by both eigenvalue algorithms to terminate computations if all eigenvalues below the specified frequency have been found.

The DETERMINANT SEARCH algorithm computes eigenvalues in order from "1" to "NF". If the Nth eigenvalue ($1 \le N < NF$) has a frequency greater than "COFQ", the remaining (NF-N) eigenvalues are not computed.

A. MODE SHAPES AND FREQUENCIES (continued)

The SUBSPACE ITERATION algorithm terminates calculation when the Nth eigenvalue is accurate (i.e., does not change with iteration) to a tolerance of RTOL. As before, the Nth eigenvalue is the nearest eigenvalue higher than COFQ. If the SUBSPACE ITERATION solution determines N eigenvalues less than COFQ (where, N <NF) , the STURM SEQUENCE check (if requested) is performed using the Nth (rather than the NFth) eigenvalue as a shift.

Only those modes whose frequencies are less than COFQ will be used in the TIME HISTORY or RESPONSE SPECTRUM analyses (Sections VII.B and C, below).

(6) The starting iteration vectors, together with control information, must be written onto TAPE10 before the program execution is started. Appendix B describes the creation of TAPE10 and gives the required control cards.

(7) The program does not calculate rigid body modes, i.e. the system must have been restraint so that no rigid body modes are present. In exact arithmetic the element d_{nn} of the matrix D in the triangular factorization of the stiffness matrix, i.e. $K = LDL^T$, is zero if a rigid body mode is present. In computer arithmetic the element d_{nn} is small when compared with the other elements of the matrix D. If this condition occurs the program stops with a message.

Note: If many "artificially" stiff boundary elements are used, the average of the elements of D will be artificially large. Consequently, d_{nn} may be small in comparison, and although no rigid body modes may be present, the program will stop. In a dynamic analysis it is recommended not to use very stiff boundary elements.

END OF DATA CASE INPUT (NDYN.EQ.1)

B. RESPONSE HISTORY ANALYSIS (NDYN.EQ.2 or NDYN.EQ.4)

The NDYN.EQ.2 option uses the ("NF") mode shapes and frequencies computed in the preceeding Section (VII.A) to perform a mode superposition solution for forced response. The NDYN.EQ.4 option initiates a direct step-by-step integration of the coupled system equations, i.e. no eigenvalue solution has been performed and no transformation to the eigenvector basis is now carried out. The data input is identical to the case NDYN.EQ.2 except for the definition of damping. Dynamic response can be produced by two (2) general types of forcing function:

(1) ground acceleration input in any (or all) of the three (3) global (X,Y,Z) directions;

and/or

(2) time varying loads (forces/moments) applied in any (or all) nodal degrees of freedom (except - "slave" - degrees of freedom)

Time dependent forcing functions (whether loads or ground acceleration components) are described in two steps. First, a number (1 or more are possible) of non-dimensional time functions are specified tabularly by a set of descrete points: $[f(t_i),t_i]$, where $i = 1,2,\ldots,k$. Each different time function may have a different number of definition points (k). A particular forcing function applied at some point on the structure is then defined by a scalar multiplier ("β", say) and reference to one of the input time functions ("f(t)", say). The actual force (or acceleration) at any time ("τ", say) equals $\beta \times f(\tau)$; $f(\tau)$ is found by linear interpolation between two of the input time points $\{t_i,t_{i+1}\}$, where $t_i \le \tau \le t_{i+1}$.

Assuming that the solution begins at time zero (0), an independent arrival time (t_a, where $t_a \ge 0$) may be assigned to each forcing function. The forcing function is not applied to the system until the solution time ("τ", say) equals the arrival time, t_a. Interpolation for function values is based on relative time within the function table; i.e., $g(\tau) = f(\tau - t_a)$.

The structure is assumed to be at rest at time zero; i.e., zero initial displacements and velocities are assumed at time of solution start.

The following data are required for a Forced Dynamic Response Analysis:

1. Control Card (5I5,2F10.0)

notes	columns	variable	entry
(1)	1 - 5	NFN	Number of different time functions; GE.1

B. RESPONSE HISTORY ANALYSIS (continued)

notes	columns	variable	entry
(2)	6 - 10	NGM	Ground motion indicator; EQ.0; no ground motion is input EQ.1; read ground motion control card (Section VII.B.3)
(3)	11 - 15	NAT	Number of different arrival times for the forcing functions; EQ.0; all arrival times are zero
(4)	16 - 20	NT	Total number of solution time steps; GE.1
(5)	21 - 25	NOT	Output print interval for stresses, displacements, etc. GE.1 and LE.NT
(4)	26 - 35	DT	Solution time step, Δt; GT.0
(6)	36 - 45	DAMP	Damping factor to be applied to all NF modes (fraction of critical); GE.0

In case of NDYN.EQ.4 use

(6)	36 - 45	ALPHA	Damping factor α
(7)	46 - 55	BETA	Damping factor β

NOTES/

(1) At least one (1) time function must be input.

(2) If no ground acceleration acts on the structure, set "NGM" to zero and skip Section VII.B.3, below. Both ground acceleration and nodal force input are allowed.

(3) If no arrival time values are input, all forcing functions begin acting on the structure at time zero. The same arrival time value may be referenced by different forcing functions. "NAT" determines the number of non-zero entries that the program expects to read in Section VII.B.4, below.

(4) The program performs a step-by-step integration of the equations of motion using a scheme which is unconditionally stable with respect to time step size, Δt. In case NDYN.EQ.2 the modal uncoupled equations of motion are integrated. In case NDYN.EQ.4 the coupled system equations are integrated. If "T" is the period of the highest numbered mode (normally the NFth mode) that is to be included in the response calculation, Δt should be chosen such that $\Delta t/T < 0.1$. A

B. RESPONSE HISTORY ANALYSIS (continued)

larger time step (i.e., $\Delta t > 0.1T$) will not cause failure (instability), but participation of the higher modes is "filtered" from the predicted response. In general, with increasing time step size the solution is capable of capturing less of the higher frequency participation.

(5) The program computes system displacements at every solution time step, but printing of displacements and recovery of element stresses is only performed at solution step intervals of "NOT". NOT must be at least "1" and is normally selected in the range of 10 to 100.

(6) The damping factor ("DAMP") is applied to all NF modes. The admissible range for DAMP is between 0.0 (no damping) and 1.0 (100% of critical viscous damping).

(7) In case NDYN.EQ.4 the damping matrix used is $C = \alpha M + \beta K$, where α and β are defined in columns 36 to 55.

B. RESPONSE HISTORY ANALYSIS (continued)

2. Time-Varying Load Cards (4I5,F10.0)

notes	columns	variable	entry
(1)	1 - 5	NP	Nodal point number where the load component (force or moment) is applied; GE.1 and LE.NUMNP EQ.0 last card only
(2)	10	IC	Degree of freedom number; GE.1 and LE.6 (δX=1, δY=2, δZ=3, ΦX=4, ΦY=5, ΦZ=6)
(3)	11 - 15	IFN	Time function number; GE.1 and LE.NTFN
(4)	16 - 20	IAT	Arrival time number; EQ.0; load applied at solution start GE.1; non-zero arrival time
(5)	21 - 30	P	Scalar multiplier for the time function; EQ.0; no load applied

NOTES:

(1) One card is required for each nodal degree of freedom having applied time varying loads. Cards must be input in ascending node point order. This sequence of cards must be terminated with a blank card. A blank card must be supplied even if no loads are applied to the system.

(2) The same node may have more than one degree of freedom loaded; arrange degrees of freedom references ("IC") in ascending sequence at any given node.

(3) A non-zero time function number ("IFN") must be given for each forcing function. IFN must be between 1 and NFN. The time functions are input tabularly in Section VII.B.5, below. Function values at times between input time points are computed with linear interpolation.

(4) If "IAT" is zero (or blank), the forcing function is assumed to act on the system beginning at time zero. If IAT is input as a positive integer between 1 and NAT, the IATth arrival time (defined in Section VII.B.4, below) is used to delay the application of the forcing function; i.e., the forcing function begins acting on the structure when the solution reaches the IATth arrival time value.

(5) The actual magnitude of force (or moment) acting on the model at time, t, equals the product: ("P") x (value of function number "IFN" at time, t).

VII. DYNAMIC ANALYSES (continued)

B. RESPONSE HISTORY ANALYSIS (continued)

3. Ground Motion Control Card (6I5)

notes	columns	variable	entry
(1)	1 - 5	NFNX	Time function number describing the ground acceleration in the X-direction
	6 - 10	NFNY	Time function number describing the ground acceleration in the Y-direction
	11 - 15	NFNZ	Time function number describing the ground acceleration in the Z-direction
(2)	16 - 20	NATX	Arrival time number, X-direction
	21 - 25	NATY	Arrival time number, Y-direction
	26 - 30	NATZ	Arrival time number, Z-direction

NOTES:

(1) This card must be input only if the ground motion indicator ("NGM") was set equal to one (1) on the Control Card (Section, VII.B.1, above). A zero time function number indicates that no ground motion is applied for that particular direction.

(2) Zero arrival time references mean that the ground acceleration (if applied) begins acting on the structure at time zero (0). Non-zero references must be integers in the range 1 to NAT.

B. RESPONSE HISTORY ANALYSIS (continued)

4. Arrival Time Cards

a. card one (8F10.0)

notes	columns	variable	entry
(1)	1 - 10	AT(1)	Arrival time number 1
	11 - 20	AT(2)	Arrival time number 2
		. . .	. . .
	71 - 80	AT(8)	Arrival time number 8

b. card two (8F10.0) - (required if NAT.GT.8)

notes	columns	variable	entry
	1 - 10	AT(9)	Arrival time number 9
		etc.	etc.

NOTES:

(1) The entry ("NAT") given in cc 11-15 on the Control Card (Section VII.B.1, above) specifies the total number of arrival time entries to be read in this section. Input as many cards as are required to define "NAT" different arrival times, eight (8) entries per card. If no arrival times were requested (NAT.EQ.0), supply one (1) blank card in this section.

B. RESPONSE HISTORY ANALYSIS (continued)

5. Time Function Definition Cards

Supply one set (card 1 and card(s) 2) of input for each of the "NFN" time functions requested in cc 1-5 of the Control Card (Section VII.B.1, above). At least one set of time function cards is expected in this section. The card sets are input in ascending function number order.

a. card 1 (I5,F10.0,12A5)

notes	columns	variable	entry
(1)	1 - 5	NLP	Number of function definition points; GE.2
(2)	6 - 15	SFTR	Scale factor to be applied to f(t) values; EQ.0; default set to "1.0"
	16 - 75	HED(12)	Label information (to be printed with output) describing this function table

NOTES:

(1) At least two points (i.e., 2 pairs: $f(t_i), t_i$) must be specified for each time function. Less than two points would preclude linear interpolation in the table for f(t).

(2) The scale factor "SFTR" is used to multiply function values only; i.e., input time values are not changed. If the scale factor is omitted, SFTR is re-set by the program to "1.0" thereby leaving input function values unchanged.

B. RESPONSE HISTORY ANALYSIS (continued)

5. Time Function Definition Cards (continued)

b. card(s) 2 (12F6.0)

notes	columns	variable	entry
(1)	1 - 6	T(1)	Time values at point 1, t_1
	7 - 12	F(1)	Function value at point 1, $f(t_1)$
	13 - 18	T(2)	Time value at point 2, t_2
	19 - 24	F(2)	Function value at point 2, $f(t_2)$
		etc.	etc.

NOTES

(1) Input as many card(s) 2 as are required to define "NLP" pairs of $t_i, f(t_i)$, six (6) pairs per card. Pairs must be input in order of ascending time value. Time at point one must be zero, and care must be taken to ensure that the highest (last) input time value (t_{NLP}) is at least equal to the value of time at the end of solution; i.e., the time span for all functions must cover the solution time period otherwise the interpolation for function values will fail. For the case of non-zero arrival times associated with a particular function, the shortest arrival time reference ("t_A", say) plus (+) the last function time ("t_{NLP}")must at least equal the time at the end of the solution period (t_{END}, say); i.e., $t_A + t_{NLP} \geq t_{END}$.

VII. DYNAMIC ANALYSES (continued)

B. RESPONSE HISTORY ANALYSIS (continued)

6. Output Definition Cards

To minimize the amount of output which would be produced by the program if all displacements, stresses, etc. were printed, output requests for specific components must be given in this section. Time histories for selected components appear in tables; the solution step output printing interval is specified as "NOT" which is given in cc 21-25 of the Control Card (Section VII.B.1, above).

a. displacement output requests

(1) control card (2I5)

notes	columns	variable	entry
(1)	1 - 5	KKK	Output type indicator; EQ.1; print histories and maxima EQ.2; printer plot histories and recovery of maxima EQ.3; recover maxima only
(2)	6 - 10	ISP	Printer plot spacing indicator

NOTES.

(1) The type of output to be produced by the program applies to all displacement requests. KKK.EQ.0 is illegal.

(2) "ISP" controls the vertical (down the page) spacing for printer plots. Output points are printed on every (ISP+1)th line. The horizontal (across the page) width of printer plots is a constant ten (10) inches (100 print positions). ISP is used only if KKK.EQ.2.

VII. DYNAMIC ANALYSES (continued)

B. RESPONSE HISTORY ANALYSIS (continued)

6. Output Definition Cards

a. displacement output requests (continued)

(2) node displacement request cards (7I5)

notes	columns	variable	entry
(1)	1 - 5	NP	Node number GE.1 and LE.NUMNP EQ.0 last card only
(2)	6 - 10	IC(1)	Displacement component, request 1
	11 - 15	IC(2)	Displacement component, request 2
	16 - 20	IC(3)	Displacement component, request 3
	21 - 25	IC(4)	Displacement component, request 4
	26 - 30	IC(5)	Displacement component, request 5
	31 - 36	IC(6)	Displacement component, request 6 GE.1 and LE.6 EQ.0 terminates requests for the node

NOTES/

(1) Only those nodes at which output is to be produced (or at which maxima are to be determined) are entered in this section. Cards must be input in ascending node number order. Node numbers may not be repeated. This section must be terminated with a blank card.

(2) Displacement component requests ("IC") range from 1 to 6, where $1=\delta X, 2=\delta Y, 3=\delta Z, 4=\Phi X, 5=\Phi Y, 6=\Phi Z$. The first zero (or blank) encountered while reading IC(1),IC(2),...,IC(6) terminates information for the card. Displacement components at a node may be requested in any order. As an example, suppose that δY, ΦX and ΦZ are to be output at node 34; the card could be written as /34,2,4,6,0/, or /34,6,4,2,0/, etc. but only four (4) fields would have non-zero entries.

VII. DYNAMIC ANALYSES (continued)

B. RESPONSE HISTORY ANALYSIS (continued)

6. Output Definition Cards

b. element stress component output requests

(1) control card (2I5)

notes	columns	variable	entry
(1)	1 - 5	KKK	Output type indicator; EQ.1; print histories and maxima EQ.2; printer plot of histories and recovery of maxima EQ.3; recover maxima only
	6 - 10	ISP	Plot spacing indicator

NOTES

(1) See Section VII.B.6.a.(1), above.

(2) element stress component request cards (13I5)

Requests are grouped by element type; "NELTYP" groups must be input. A group consists of a series of element stress component request cards terminated by a blank card. Element number references within an element type (TRUSS, say) grouping must be in ascending order. Element number references may be omitted but not repeated. The program processes element groups in the same order as originally input in the Element Data (Section IV, above). If no output is to be produced for an element type, then input one blank card for its group.

notes	columns	variable	entry
(1)	1 - 5	NEL	Element number GE.1 EQ.0; last card in the group only
(2)	6 - 10	IS(1)	Stress component number for output, request 1
	11 - 15	IS(2)	Stress component number for output, request 2
	. . . .	. . .	. . .
	61 - 65	IS(12)	Stress component number for output, request 12

NOTES/

(1) Terminate each different element output group (type) with a blank card. Elements within a group must be in element number order (ascending); element number repetitions are illegal.

(2) The first zero (or blank) request encountered while reading IS(1), IS(2),..., IS(12) terminates information for the card. No more than twelve (12) different components may be output for any one of the elements. Table VII.1 lists the stress component numbers and corresponding descriptions for the various element types. Some element types (TRUSS, for example) have fewer than 12 components defined; only the stress component numbers listed in Table VII.1 are legal references.

END OF DATA CASE INPUT (NDYN.EQ.2 or NDYN.EQ.4)

TABLE VII.1

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	D E S C R I P T I O N
1. TRUSS	(2)	(1)	(P/A)	AXIAL STRESS
		(2)	(P)	AXIAL FORCE

* *

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	D E S C R I P T I O N
2. BEAM	(12)	(1)	(P1(I))	1-FORCE AT END I
		(2)	(V2(I))	2-SHEAR AT END I
		(3)	(V3(I))	3-SHEAR AT END I
		(4)	(T1(I))	1-TORQUE AT END I
		(5)	(M2(I))	2-MOMENT AT END I
		(6)	(M3(I))	3-MOMENT AT END I
		(7)	(P1(J))	1-FORCE AT END J
		(8)	(V2(J))	2-SHEAR AT END J
		(9)	(V3(J))	3-SHEAR AT END J
		(10)	(T1(J))	1-TORQUE AT END J
		(11)	(M2(J))	2-MOMENT AT END J
		(12)	(M3(J))	3-MOMENT AT END J

* *

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	D E S C R I P T I O N
3. PLANE-STRESS/ PLANE-STRAIN				
4. AXISYM-METRIC	(20)	(1)	(11-S0)	V- STRESS AT POINT 0
		(2)	(22-S0)	U- STRESS AT POINT 0
		(3)	(33-S0)	T- STRESS AT POINT 0
		(4)	(12-S0)	UV-STRESS AT POINT 0
		(5)	(11-S1)	V- STRESS AT POINT 1
		(6)	(22-S1)	U- STRESS AT POINT 1
		(7)	(33-S1)	T- STRESS AT POINT 1
		(8)	(12-S1)	UV-STRESS AT POINT 1
		(9)	(11-S2)	V- STRESS AT POINT 2
		(10)	(22-S2)	U- STRESS AT POINT 2
		(11)	(33-S2)	T- STRESS AT POINT 2
		(12)	(12-S2)	UV-STRESS AT POINT 2
		(13)	(11-S3)	V- STRESS AT POINT 3
		(14)	(22-S3)	U- STRESS AT POINT 3
		(15)	(33-S3)	T- STRESS AT POINT 3
		(16)	(12-S3)	UV-STRESS AT POINT 3

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	D E S C R I P T I O N
		(17)	(V -S4)	V- STRESS AT POINT 4
		(18)	(U -S4)	U- STRESS AT POINT 4
		(19)	(T -S4)	T- STRESS AT POINT 4
		(20)	(UV-S4)	UV-STRESS AT POINT 4

* *

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	DESCRIPTION
5. EIGHT NODE BRICK	(12)	(1)	(XX-SL1)	XX-STRESS AT LOCATION 1
		(2)	(YY-SL1)	YY-STRESS AT LOCATION 1
		(3)	(ZZ-SL1)	ZZ-STRESS AT LOCATION 1
		(4)	(XY-SL1)	XY-STRESS AT LOCATION 1
		(5)	(YZ-SL1)	YZ-STRESS AT LOCATION 1
		(6)	(ZX-SL1)	ZX-STRESS AT LOCATION 1
		(7)	(XX-SL2)	XX-STRESS AT LOCATION 2
		(8)	(YY-SL2)	YY-STRESS AT LOCATION 2
		(9)	(ZZ-SL2)	ZZ-STRESS AT LOCATION 2
		(10)	(XY-SL2)	XY-STRESS AT LOCATION 2
		(11)	(YZ-SL2)	YZ-STRESS AT LOCATION 2
		(12)	(ZX-SL2)	ZX-STRESS AT LOCATION 2

* *

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	DESCRIPTION
6. PLATE/ SHELL	(6)	(1)	(XX-S/R)	XX-STRESS RESULTANT
		(2)	(YY-S/R)	YY-STRESS RESULTANT
		(3)	(XY-S/R)	XY-STRESS RESULTANT
		(4)	(XX-M/R)	XX-MOMENT RESULTANT
		(5)	(YY-M/R)	YY-MOMENT RESULTANT
		(6)	(XY-M/R)	XY-MOMENT RESULTANT

* *

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	DESCRIPTION
7. BOUN- DARY	(2)	(1)	(BDRY-F)	BOUNDARY FORCE
		(2)	(BDRY-M)	BOUNDARY MOMENT

* *

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	DESCRIPTION
8. THICK SHELL AND 3-DIM.	(42)	(1)	(SXX(0))	XX-STRESS AT CENTROID (0)
		(2)	(SYY(0))	YY-STRESS AT CENTROID (0)
		(3)	(SZZ(0))	ZZ-STRESS AT CENTROID (0)
		(4)	(SXY(0))	XY-STRESS AT CENTROID (0)
		(5)	(SYZ(0))	YZ-STRESS AT CENTROID (0)
		(6)	(SZX(0))	ZX-STRESS AT CENTROID (0)
		(7)	(SXX(1))	XX-STRESS AT CENTER OF FACE 1

ELEMENT TYPE	MAXIMUM NUMBER OF COMPONENTS	STRESS COMPONENT NUMBER	OUTPUT SYMBOL	D E S C R I P T I O N
		(8)	(SYY(1))	YY-STRESS AT CENTER OF FACE 1
		(9)	(SZZ(1))	ZZ-STRESS AT CENTER OF FACE 1
		(10)	(SXY(1))	XY-STRESS AT CENTER OF FACE 1
		(11)	(SYZ(1))	YZ-STRESS AT CENTER OF FACE 1
		(12)	(SZX(1))	ZX-STRESS AT CENTER OF FACE 1
		(13)	(SXX(2))	XX-STRESS AT CENTER OF FACE 2
		(14)	(SYY(2))	YY-STRESS AT CENTER OF FACE 2
		(15)	(SZZ(2))	ZZ-STRESS AT CENTER OF FACE 2
		(16)	(SXY(2))	XY-STRESS AT CENTER OF FACE 2
		(17)	(SYZ(2))	YZ-STRESS AT CENTER OF FACE 2
		(18)	(SZX(2))	ZX-STRESS AT CENTER OF FACE 2
		(19)	(SXX(3))	XX-STRESS AT CENTER OF FACE 3
		(20)	(SYY(3))	YY-STRESS AT CENTER OF FACE 3
		(21)	(SZZ(3))	ZZ-STRESS AT CENTER OF FACE 3
		(22)	(SXY(3))	XY-STRESS AT CENTER OF FACE 3
		(23)	(SYZ(3))	YZ-STRESS AT CENTER OF FACE 3
		(24)	(SZX(3))	ZX-STRESS AT CENTER OF FACE 3
		(25)	(SXX(4))	XX-STRESS AT CENTER OF FACE 4
		(26)	(SYY(4))	YY-STRESS AT CENTER OF FACE 4
		(27)	(SZZ(4))	ZZ-STRESS AT CENTER OF FACE 4
		(28)	(SXY(4))	XY-STRESS AT CENTER OF FACE 4
		(29)	(SYZ(4))	YZ-STRESS AT CENTER OF FACE 4
		(30)	(SZX(4))	ZX-STRESS AT CENTER OF FACE 4
		(31)	(SXX(5))	XX-STRESS AT CENTER OF FACE 5
		(32)	(SYY(5))	YY-STRESS AT CENTER OF FACE 5
		(33)	(SZZ(5))	ZZ-STRESS AT CENTER OF FACE 5
		(34)	(SXY(5))	XY-STRESS AT CENTER OF FACE 5
		(35)	(SYZ(5))	YZ-STRESS AT CENTER OF FACE 5
		(36)	(SZX(5))	ZX-STRESS AT CENTER OF FACE 5
		(37)	(SXX(6))	XX-STRESS AT CENTER OF FACE 6
		(38)	(SYY(6))	YY-STRESS AT CENTER OF FACE 6
		(39)	(SZZ(6))	ZZ-STRESS AT CENTER OF FACE 6
		(40)	(SXY(6))	XY-STRESS AT CENTER OF FACE 6
		(41)	(SYZ(6))	YZ-STRESS AT CENTER OF FACE 6
		(42)	(SZX(6))	ZX-STRESS AT CENTER OF FACE 6

* *

9. PIPE

A. TANGENT (12)	(1)	(PX(I))	X-FORCE AT END I
	(2)	(VY(I))	Y-SHEAR AT END I
	(3)	(VZ(I))	Z-SHEAR AT END I
	(4)	(TX(I))	X-TORQUE AT END I
	(5)	(MY(I))	Y-MOMENT AT END I
	(6)	(MZ(I))	Z-MOMENT AT END I
	(7)	(PX(J))	X-FORCE AT END J
	(8)	(VY(J))	Y-SHEAR AT END J
	(9)	(VZ(J))	Z-SHEAR AT END J
	(10)	(TX(J))	X-TORQUE AT END J
	(11)	(MY(J))	Y-MOMENT AT END J
	(12)	(MZ(J))	Z-MOMENT AT END J
B. BEND (18)	(1)	(PX(I))	X-FORCE AT END I
	(2)	(VY(I))	Y-SHEAR AT END I
	(3)	(VZ(I))	Z-SHEAR AT END I
	(4)	(TX(I))	X-TORQUE AT END I
	(5)	(MY(I))	Y-MOMENT AT END I
	(6)	(MZ(I))	Z-MOMENT AT END I
	(7)	(PX(C))	X-FORCE AT CENTER OF ARC
	(8)	(VY(C))	Y-SHEAR AT CENTER OF ARC
	(9)	(VZ(C))	Z-SHEAR AT CENTER OF ARC
	(10)	(TX(C))	X-TORQUE AT CENTER OF ARC
	(11)	(MY(C))	Y-MOMENT AT CENTER OF ARC
	(12)	(MZ(C))	Z-MOMENT AT CENTER OF ARC
	(13)	(PX(J))	X-FORCE AT END J
	(14)	(VY(J))	Y-SHEAR AT END J
	(15)	(VZ(J))	Z-SHEAR AT END J
	(16)	(TX(J))	X-TORQUE AT END J
	(17)	(MY(J))	Y-MOMENT AT END J
	(18)	(MZ(J))	Z-MOMENT AT END J

* *
* *

VII. DYNAMIC ANALYSES (continued)

C. RESPONSE SPECTRUM ANALYSIS (NDYN.EQ.3)

This option combines all (NF) mode shapes and frequencies computed during the eigenvalue solution (Section VII.A) to calculate R.M.S. stresses/deflections due to an input displacement (or acceleration) spectrum. The input spectrum is applied in varying proportions in the global X,Y,Z directions. For the case of a non-zero cut-off frequency "COFQ" (Section VII.A), only those modes whose frequencies are less than COFQ will be combined in the R.M.S. analysis.

1. Control Card (3F10.0,I5)

notes	columns	variable	entry
(1)	1 - 10	FX	Factor for X-direction input
	11 - 20	FY	Factor for Y-direction input
	21 - 30	FZ	Factor for Z-direction input EQ.0; not acting
(2)	31 - 35	IST	Input spectrum type; EQ.0; displacement vs. period EQ.1; acceleration vs. period

NOTES/

(1) All three (3) direction factors may be non-zero in which case the entries represent the X,Y,Z components of the input direction vector.

(2) "IST" defines the type of spectrum table to be input immediately following. The spectral displacements ("S_d") and accelerations ("S_a") are assumed to be related as follows: $S_a = (4\pi^2 f^2)(S_d)$.

C. RESPONSE SPECTRUM ANALYSIS (continued)

2. Spectrum Cards

a. heading card (12A6)

notes	columns	variable	entry
	1 - 72	HED(12)	Heading information used to label the spectrum table

b. control card (I5,F10.0)

notes	columns	variable	entry
	1 - 5	NPTS	Number of definition points in the spectrum table; GE.2
	6 - 15	SFTR	Scale factor used to adjust the displacement (or acceleration) ordinates in the spectrum table EQ.1.0; no adjustment

c. spectrum data (2F10.0)

notes	columns	variable	entry
(1)	1 - 10	T	Period (reciprocal of frequency)
(2)	11 - 20	S	Value of displacement (or acceleration if IST.EQ.1)

NOTES/

(1) Input one definition point per card; "NPTS" cards are required in this section. Cards must be arranged in ascending value of period.

(2) "S" is interpreted to be a displacement quantity if "IST" was input as zero. For IST.EQ.1, "S" is an acceleration value.

END OF DATA CASE INPUT (NDYN.EQ.3)

APPENDIX A - CONTROL CARDS AND DECK SET-UP FOR DYNAMIC ANALYSIS RE-START

The purpose of this appendix is to describe the procedure (including control cards and deck set-up) required for program re-start following an eigenvalue/eigenvector extraction analysis. The re-start option has been included in the program in order to make a repeated forced response or spectrum analysis possible without solving each time for the required eigensystem. For medium-to-large size models, eigenvalue solution is quite costly when compared to the forced response calculations; hence, excessive costs may be incurred if the entire job has to be re-run due to improper specification of forcing functions or input spectra, inadequate requests, etc. For small models (less than 100 nodes, say) the extra effort required for re-start is normally not justified.

A complete dynamic analysis utilizing the re-start feature requires that the job be run in two (2) steps:

JOB(1): Eigenvalue extraction solution only, after which program files TAPE1,TAPE2,TAPE7,TAPE8, and TAPE9 are saved on the re-start tape.

JOBS(2): Re-instatement of program files TAPE1,TAPE2,TAPE7,TAPE8, and TAPE9 from the re-start tape followed by a Dynamic Response Analysis (NDYN.EQ.-2) or a Response Spectrum Analysis (NDYN.EQ.-3).

For a given model, the first job [JOB(1)] creating the re-start tape is run only once. The re-start tape then contains all the initial information required by the program at the beginning of a forced response analysis. More than one second job [JOBS(2)] may be run using the re-start tape as initial input; i.e., the re-start tape is not destroyed.

Control cards and deck set-up for execution on the CDC 6400 computer at the University of California, Berkeley are given below:

JOB(1) - EIGENVALUE SOLUTION/RE-START TAPE CREATION

Notes	Card Deck
(1)	Job number, 1, 200, 120000,300. User Name
(2)	REQUEST, TP1,I. Reel No., Tape User Name
(3)	CØPYBF, TP1,SAP4
	UNLØAD,TP1
(4)	LGØ,SAP4
	REWIND,TAPE1,TAPE2,TAPE7,TAPE8,TAPE9
(5)	REQUEST,RESTART,I. Reel No., Tape User Name, ØUTPUT
(6)	CØPYBF,TAPE1,RESTART
	CØPYBF,TAPE2,RESTART
	CØPYBF,TAPE7,RESTART
	CØPYBF,TAPE8,RESTART
	CØPYBF,TAPE9,RESTART
(7)	7-8-9

PROBLEM DATA DECK:

- I. HEADING CARD
- II. MASTER CONTROL CARD with
  - (LL.EQ.0)
  - (NF.GE.1)
  - (NDYN.EQ.1)
  - (MØDEX.EQ.0)
- III. JOINT DATA
- IV. ELEMENT DATA
- V. CONCENTRATED MASS DATA
- VI. ELEMENT LOAD MULTIPLIERS
- VII. DYNAMIC ANALYSIS
  - A. Mode Shapes and Frequencies

blank card

blank card

(8) 6-7-8-9

NOTES:

(1) The job control card parameters are defined as follows:
- "1" = Number of tape drives required for the job.
- "200" = CPU time limit (in octal seconds).
- "120000" = Central memory field length (in octal).
- "300" = Page limit for printing.

(2) Tape containing binary version of program (TP1) is requested.

(3) Binary version of the program is copied onto a disk file (SAP4).

(4) Program is loaded and execution is initiated.

(5) A blank tape (RESTART) is requested.

(6) The contents of disk files TAPE1,TAPE2, etc. are copied onto tape RESTART.

(7) End-of-record card: 7,8,9 punched in column 1.

(8) End-of-file card: 6,7,8,9 punched in column 1.

JOB (2) - RE-START FOR RESPONSE HISTORY ANALYSIS (NDYN.EQ.-2)
or RESPONSE SPECTRUM ANALYSIS (NDYN.EQ.-3)

```
Notes     Card Deck

          Job number, 1,200,120000,300.   User Name
          REQUEST, RESTART, I.   Reel No., User Name
          CØPYBF,RESTART,TAPE1
          CØPYBF,RESTART,TAPE2
 (1)      CØPYBF,RESTART,TAPE7
          CØPYBF,RESTART,TAPE8
          CØPYBF,RESTART,TAPE9
          REWIND,TAPE1,TAPE2,TAPE7,TAPE8,TAPE9
          UNLØAD,RESTART
          REQUEST,TP1,I.  Reel No., User Name
 (2)      CØPYBF,TP1,SAP4
          LGØ,SAP4
          7-8-9

          PROBLEM DATA DECK
                I.     HEADING CARD
                II.    MASTER CONTROL CARD with
                       (LL.EQ.0)
                       (NF.GE.1)
                       (NDYN.EQ.-2 or -3)
 (3)                   (MODEX.EQ.0)
                VII.   DYNAMIC ANALYSIS
                       B.    Dynamic Response Analysis (NDYN.EQ.-2)
                or
                       C.    Response Spectrum Analysis (NDYN.EQ.-3)
          blank card
          blank card

          6-7-8-9
```

NOTES/

(1) The disk files TAPE1,TAPE2, etc. are re-created using the information saved on tape RESTORE.

(2) The binary version of the program is again obtained from tape TP1.

(3) Normally, the number of frequencies ("NF") entered on the MASTER CONTROL CARD for a re-start case has the same value as was specified earlier when the eigenvalue problem was solved in JOB(1). If a value for the cut-off frequency ("COFQ") was entered on the "Mode Shapes and Frequencies" control card [in JOB(1)] and the program extracted fewer than "NF" frequencies (eigenvalues), then only the actual number of eigenvalues computed by the program in JOB(1) is specified for "NF" in this re-start run.

APPENDIX B: CONTROL CARDS AND DECK SET-UP FOR USE OF STARTING ITERATION VECTORS

In the dynamic analysis of large-order systems, the solution of the required eigensystem is normally the most expensive phase. The option described in this appendix demonstrates how it is possible to use NFØ previously calculated eigenvalues and vectors when the solution for NF ≥ NFØ eigenvalues and eigenvectors is required.

Assume that in Job(1), the solution for NFØ eigenvalues and eigenvectors was performed. At the end of this job, TAPE2 and TAPE7 must have been saved on a physical tape, say "RESTART". Assuming that in JOB(2) the solution of NF eigenvalues and eigenvectors is required, then prior to the execution of this job, tape RESTART needs to be copied onto TAPE10.

This procedure was performed with the following control cards on the CDC 6400 of the University of California at Berkeley:

JOB(1) - SOLUTION FOR NFØ EIGENVALUES/RESTART TAPE CREATION

Notes	Card Deck
(1)	Job No., 1,200,120000,500. User Name REQUEST,TP1,I. Reel No., Tape User Name CØPYBF,TP1,SAP4 UNLØAD,TP1
(2)	REQUEST,TAPE2,NB REQUEST,TAPE7,NB
	LGØ,SAP4
	REWIND,TAPE2,TAPE7
(3)	REQUEST,RESTART,I. Reel No.,Tape User Name, OUTPUT
(4)	CØPYBR,TAPE2,RESTART,1 CØPYBF,TAPE7,TP3
	7-8-9
	PROBLEM DATA DECK
	6-7-8-9

Notes:

(1) See Notes (1) - (4) in Appendix A.

(2) The computer is directed to write on disk files TAPE2 and TAPE7 in an unblocked format.

(3) A blank tape (RESTART) is requested onto which the contents of files TAPE2 and TAPE7 are to be written.

(4) The contents of files TAPE2 and TAPE7 are written as one file onto tape RESTART.

JOB(2) - SOLUTION FOR ADDITIONAL EIGENVALUES USING THE INFORMATION STORED ON TAPE "RESTART"

Notes	Card Deck
	Job No.,1,200,120000,500. User Name
(1)	REQUEST,RESTART,I. Reel No., Tape User Name
	REQUEST,TAPE10,NB
	REQUEST,TAPE2,NB
	REQUEST,TAPE7,NB
(2)	CØPYBF,RESTART,TAPE10
	UNLOAD,RESTART
	REWIND,TAPE10
(3)	REQUEST,TP1,I. Reel No., Tape User Name
	CØPYBF,TP1,SAP4
	LGØ,SAP4
	7-8-9
	PROGRAM DATA DECK
	6-7-8-9

Notes/

(1) TAPE10 (as TAPE2 and TAPE7 if they are to be used for further restarts,) is requested to be an unblocked file.

(2) The contents of tape RESTART are copied into TAPE10 as one file.

(3) Program execution.

EARTHQUAKE ENGINEERING RESEARCH CENTER REPORTS

EERC 67-1 "Feasibility Study Large-Scale Earthquake Simulator Facility", by J. Penzien, J. G. Bouwkamp, R. W. Clough and D. Rea - 1967 (PB 187 905)

EERC 68-1 Unassigned

EERC 68-2 "Inelastic Behavior of Beam-to-Column Subassemblages Under Repeated Loading", by V. V. Bertero - 1968 (PB 184 888)

EERC 68-3 "A Graphical Method for Solving the Wave Reflection-Refraction Problem", by H. D. McNiven and Y. Mengi - 1968 (PB 187 943)

EERC 68-4 "Dynamic Properties of McKinley School Buildings", by D. Rea, J. G. Bouwkamp and R. W. Clough - 1968 (PB 187 902)

EERC 68-5 "Characteristics of Rock Motions During Earthquakes", by H. B. Seed, I. M. Idriss and F. W. Kiefer - 1968 (PB 188 338)

EERC 69-1 "Earthquake Engineering Research at Berkeley" - 1969 (PB 187 906)

EERC 69-2 "Nonlinear Seismic Response of Earth Structures", by M. Dibaj and J. Penzien - 1969 (PB 187 904)

EERC 69-3 "Probabilistic Study of the Behavior of Structures During Earthquakes", by P. Ruiz and J. Penzien - 1969 (PB 187 886)

EERC 69-4 "Numerical Solution of Boundary Value Problems in Structural Mechanics by Reduction to an Initial Value Formulation", by N. Distefano and J. Schujman - 1969 (PB 187 942)

EERC 69-5 "Dynamic Programming and the Solution of the Biharmonic Equation", by N. Distefano - 1969 (PB 187 941)

EERC 69-6 "Stochastic Analysis of Offshore Tower Structures", by A. K. Malhotra and J. Penzien - 1969 (PB 187 903)

EERC 69-7 "Rock Motion Accelerograms for High Magnitude Earthquakes", by H. B. Seed and I. M. Idriss - 1969 (PB 187 940)

EERC 69-8 "Structural Dynamics Testing Facilities at the University of California, Berkeley", by R. M. Stephen, J. G. Bouwkamp, R. W. Clough and J. Penzien - 1969 (PB 189 111)

Note: Numbers in parentheses are Accession Numbers assigned by the National Technical Information Service. Copies of these reports may be ordered from the National Technical Information Service, Springfield, Virginia, 22151. Either the accession number or a complete citation should be quoted on orders for the reports.

Revised 4/23/73

EERC 69-9 "Seismic Response of Soil Deposits Underlain by Sloping Rock Boundaries", by H. Dezfulian and H. B. Seed - 1969 (PB 189 114)

EERC 69-10 "Dynamic Stress Analysis of Axisymmetric Structures Under Arbitrary Loading", by S. Ghosh and E. L. Wilson - 1969 (PB 189 026)

EERC 69-11 "Seismic Behavior of Multistory Frames Designed by Different Philosophies", by J. C. Anderson and V. V. Bertero - 1969 (PB 190 662)

EERC 69-12 "Stiffness Degradation of Reinforcing Concrete Structures Subjected to Reversed Actions", by V. V. Bertero, B. Bresler and H. Ming Liao - 1969 (PB 202 942)

EERC 69-13 "Response of Non-Uniform Soil Deposits to Travel Seismic Waves", by H. Dezfulian and H. B. Seed - 1969 (PB 191 023)

EERC 69-14 "Damping Capacity of a Model Steel Structure", by D. Rea, R. W. Clough and J. G. Bouwkamp - 1969 (PB 190 663)

EERC 69-15 "Influence of Local Soil Conditions on Building Damage Potential During Earthquakes", by H. B. Seed and I. M. Idriss - 1969 (PB 191 036)

EERC 69-16 "The Behavior of Sands Under Seismic Loading Conditions", by M. L. Silver and H. B. Seed - 1969 (AD 714 982)

EERC 70-1 "Earthquake Response of Concrete Gravity Dams", by A. K. Chopra - 1970 (AD 709 640)

EERC 70-2 "Relationships Between Soil Conditions and Building Damage in the Caracas Earthquake of July 29, 1967", by H. B. Seed, I. M. Idriss and H. Dezfulian - 1970 (PB 195 762)

EERC 70-3 "Cyclic Loading of Full Size Steel Connections", by E. P. Popov and R. M. Stephen - 1970 (PB 213 545)

EERC 70-4 "Seismic Analysis of the Charaima Building, Caraballeda, Venezuela", by Subcommittee of the SEAONC Research Committee, V. V. Bertero, P. F. Fratessa, S. A. Mahin, J. H. Sexton, A. C. Scordelis, E. L. Wilson, L. A. Wyllie, H. B. Seed, and J. Penzien, Chairman - 1970 (PB 201 455)

EERC 70-5 "A Computer Program for Earthquake Analysis of Dams", by A. K. Chopra and P. Chakrabarti - 1970 (AD 723 994)

EERC 70-6 "The Propagation of Love Waves Across Non-Horizontally Layered Structures", by J. Lysmer and L. A. Drake - 1970 (PB 197 896)

EERC 70-7 "Influence of Base Rock Characteristics on Ground Response", by J. Lysmer, H. B. Seed and P. B. Schnabel - 1970 (PB 197 897)

EERC 70-8 "Applicability of Laboratory Test Procedures for Measuring Soil Liquefaction Characteristics Under Cyclic Loading", by H. B. Seed and W. H. Peacock - 1970 (B 198 016)

EERC 70-9 "A Simplified Procedure for Evaluating Soil Liquefaction Potential", by H. B. Seed and I. M. Idriss - 1970 (PB 198 009)

EERC 70-10 "Soil Moduli and Damping Factors for Dynamic Response Analysis", by H. B. Seed and I. M. Idriss - 1970 (PB 197 869)

EERC 71-1 "Koyna Earthquake and the Performance of Koyna Dam", by A. K. Chopra and P. Chakrabarti - 1971 (AD 731 496)

EERC 71-2 "Preliminary In-Situ Measurements of Anelastic Absorption in Soils Using a Prototype Earthquake Simulator", by R. D. Borcherdt and P. W. Rodgers - 1971 (PB 201 454)

EERC 71-3 "Static and Dynamic Analysis of Inelastic Frame Structures", by F. L. Porter and G. H. Powell - 1971 (PB 210 135)

EERC 71-4 "Research Needs in Limit Design of Reinforced Concrete Structures", by V. V. Bertero - 1971 (PB 202 943)

EERC 71-5 "Dynamic Behavior of a High-Rise Diagonally Braced Steel Building", by D. Rea, A. A. Shah and J. G. Bouwkamp - 1971 (PB 203 584)

EERC 71-6 "Dynamic Stress Analysis of Porous Elastic Solids Saturated With Compressible Fluids", by J. Ghaboussi and E. L. Wilson - 1971 (PB 211 396)

EERC 71-7 "Inelastic Behavior of Steel Beam-to-Column Subassemblages", by H. Krawinkler, V. V. Bertero and E. P. Popov - 1971 (PB 211 335)

EERC 71-8 "Modification of Seismograph Records for Effects of Local Soil Conditions" by P. Schnabel, H. B. Seed and J. Lysmer - 1971 (PB 214 450)

EERC 72-1 "Static and Earthquake Analysis of Three Dimensional Frame and Shear Wall Buildings" by E. L. Wilson and H. H. Dovey - 1972 (PB 212 589)

EERC 72-2 "Accelerations in Rock For Earthquakes in the Western United States", by P. B. Schnabel and H. B. Seed - 1972 (PB 213 100)

EERC 72-3 "Elastic-Plastic Earthquake Response of Soil-Building Systems" by T. Minami and J. Penzien - 1972 (PB 214 868)

EERC 72-4 "Stochastic Inelastic Response of Offshore Towers to Strong Motion Earthquakes", by M. K. Kaul and J. Penzien - 1972 (PB 215 713)

EERC 72-5 Cyclic Behavior of Three Reinforced Concrete Flexural Members With High Shear" by E. P. Popov, V. V. Bertero and H. Krawinkler - 1972 (PB 214 555)

EERC 72-6 "Earthquake Response of Gravity Dams Including Reservoir Interaction Effects" by P. Chakrabarti and A. K. Chopra - 1972.

EERC 72-7 "Dynamic Properties of Pine Flat Dam", by D. Rea, C. Y.Liau and A. K. Chopra - 1972.

EERC 72-8 "Three Dimensional Analysis of Building Systems", by E.L. Wilson and H.H. Dovey - 1972.

EERC 72-9 "Rate of Loading Effects on Uncracked and Repaired Reinforced Concrete Members", by V.V. Bertero, D. Rea, S. Mahin and M. Atalay - 1973

EERC 72-10 "Computer Program for Static and Dynamic Analysis of Linear Structural Systems", by E.L. Wilson, K.J. Bathe, J.E. Peterson and H.H. Dovey - 1972.

EERC 72-11 "Literature Survey - Seismic Effects on Highway Bridges" by T. Iwasaki, J. Penzien and R. Clough - 1972 (PB 215 613)

EERC 72-12 "SHAKE, a Computer Program for Earthquake Response Analysis of Horizontally Layered Sites", by P.B. Schnabel and J. Lysmer - 1972.

EERC 73-1 "Optimal Seismic Design of Multistory Frames", by V.V. Bertero and H. Kamil - 1973.

EERC 73-2 "Analysis of the Slides in the San Fernando Dams During the Earthquake of February 9, 1971", by H.B. Seed, K.L. Lee, I.M. Idriss and F. Makdisi - 1973.

EERC 73-3 "Computer Aided Ultimate Load Design of Unbraced Multistory Steel Frames", by M.B. El-Hafez and G.J. Powell - 1973.

EERC 73-4 "Experimental Investigation into the Seismic Behavior of Critical Regions of Reinforced Concrete Components as Influenced by Moment and Shear", by M. Celebi and J. Penzien - 1973 (PB 215 884)

EERC 73-5 "Hysteretic Behavior of Epoxy-Repaired Reinforced Concrete Beams", by M. Celebi and J. Penzien - 1973.

EERC 73-6 "General Purpose Computer Program for Inelastic Dynamic Response of Plane Structures", by A. Kanaan and G.H. Powell - 1973.

EERC 73-7 "A Computer Program for Earthquake Analysis of Gravity Dams Including Reservoir Interaction", by P. Chakrabarti and A.K. Chopra - 1973.

EERC 73-8 "Seismic Behavior of Spandrel Frames — A Review and Outline for Future Research", by R. Razani and J.G. Bouwkamp - 1973.

EERC 73-9 "Earthquake Analysis of Structure-Foundation Systems", by A. K. Vaish and A. K. Chopra - 1973.

EERC 73-10 "Deconvolution of Seismic Response for Linear Systems", by R. B. Reimer - 1973.

EERC 73-11 "SAP IV Structure Analysis Program for Static and Dynamic Response of Linear Systems", by K. -J. Bathe, E. L. Wilson, and F. E. Peterson - 1973 (revised).

EERC 73-12 "Analytical Investigations of the Seismic Response of Tall Flexible Highway Bridges", by W. S. Tseng and J. Penzien - 1973.

EERC 73-13 "Earthquake Analysis of Multi-Story Buildings Including Foundation Interaction", by A. K. Chopra and J. A. Gutierrez - 1973 (PB 222 970).

EERC 73-14 "ADAP A Computer Program for Static and Dynamic Analysis of Arch Dams", by R. W. Clough, J. M. Raphael and S. Mojtahedi - 1973 (PB 223 763/AS).

EERC 73-15 "Cyclic Plastic Analysis of Structural Steel Joints", by R. B. Pinkney and R. W. Clough - 1973.

EERC 73-16 "QUAD-4 A Computer Program for Evaluating the Seismic Response of Soil Structures by Variable Damping Finite Element Procedures" by I. M. Idriss, J. Lysmer, R. Hwang and H. G. Seed - 1973.

EERC 73-17 "Dynamic Behavior of a Multi-Story Pyramid Shaped Building", by R. M. Stephen and J. G. Bouwkamp - 1973.

EERC 73-18 "Effect of Different Types of Reinforcing on Seismic Behavior of Short Concrete Columns", by V. V. Bertero, J. Hollings, O. Kustu, R. M. Stephen and J. G. Bouwkamp - 1973.

EERC 73-19 "Olive View Medical Center Material Studies, Phase I", by B. Bresler and V. Bertero - 1973.

EERC 73-20 "Linear and Nonlinear Seismic Analysis Computer Programs for Long Multiple-Span Highway Bridges", by W. S. Tseng and J. Penzien - 1973.

EERC 73-21 "Constitutive Models for Cyclic Plastic Deformation of Engineering Materials", by J. M. Kelly and P. P. Gillis - 1973.

EERC 73-22 "DRAIN-2D Users' Guide" by G. H. Powell - 1973.

EERC 73-23 "Earthquake Engineering at Berkeley - 1973" by D. Rea - 1973.

EERC 73-24 "Seismic Input and Structural Response During the 1971 San Fernando Earthquake" by R. B. Reimer, R. W. Clough, and J. M. Raphael - 1973.

EERC 73-25 "Earthquake Response of Axisymmetric Tower Structures Surrounded by Water", by C. Y. Liaw and A. K. Chopra - 1973.

EERC 73-26 "Investigation of the Failures of the Olive View Stairtowers During the San Fernando Earthquake and Their Implications on Seismic Design", by V. V. Bertero and Robert G. Collins - 1973.

EERC 73-27 "Further Studies on Seismic Behavior of Steel Beam-Column Subassemblages" by V. V. Bertero, H. Krawinkler and E. P. Popov - 1973.

APPENDIX E: Parallel FORTRAN Listing of PV-SAP Code

```
          Force sap of NNP ident me
          Shared integer iops(8),iopf(8)
c  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  *
C
C                                 SAP4
C                     A STRUCTURAL ANALYSIS PROGRAM
C           FOR STATIC AND DYNAMIC RESPONSE OF LINEAR SYSTEMS
C
C                K.J. BATHE , E.L. WILSON , F.E. PETERSON
C                  UNIVERSITY OF CALIFORNIA , BERKELEY
C
C          IBM CONVERSION BY UNIVERSITY OF SOUTHERN CALIFORNIA
C                             AUGUST, 1973
C                           REVISED JULY, 1974
C
C  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  **  *
C
c      IMPLICIT REAL*8(A-H,O-Z)
c       Shared REAL T,TT
       Shared REAL TT
      Shared COMMON /JUNK/HED(12),JUK(406)
      Shared COMMON /ELPAR/NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,
     &                           mtot,neq
      Shared COMMON /EM/QQQ(2846)
      Shared COMMON /DYN/IDU5(11),NDYN
      Shared COMMON /TAPES/NQQ(6)
      Shared COMMON /EXTRA/MODEX,NT8,N10SV,NT10,KEQB,NUMEL,T(10)
      Shared COMMON /SOL/NBLOCK,NEQB,LL,NF,IDUM,NEIG,NAD,NVV,ANORM,NFO
c      common /maybe/ dxx(50),dyy(50),dzz(50),ee(50),aa(50)
      Shared common /say/neqq,numee,loopur,nnblock,nterms,option
      Shared common /what/naxa(10000),irowl(10000),icolh(10000)
C
C     PROGRAM CAPACITY CONTROLLED BY THE FOLLOWING TWO STATEMENTS ...
C
      Shared COMMON /one/A(7500001)
         Shared common /time/ t1(8),t2(8),t3(8)
      Shared integer kdyn
         End declarations
      MTOT=    7500000
        Barrier
c    read option for parallel eqn solver if option is 1 then solve
c    sim. eqns by parallel subroutine if 0 solve it by original sap
        read(5,*)option
            End barrier
C
C     USE THE IBM FORTRAN EXTENDED ERROR HANDLING FACILITY TO
C     ELIMINATE PRINTOUT OF UNDERFLOW ERROR MESSAGE (ERROR NUMBER 208)
C
C***      CALL ERRSET (208,256,-1,1)
C
C
C***      CALL STIME
C
         loopur=9
       nsf=13
```

```
      NT8 = 8
       rewind 14
      REWIND NT8
      NT10= 10
      REWIND NT10
      N1=1
      rewind 13
5           zzzxg=0.
C
        Barrier
C     P R O G R A M   C O N T R O L   D A T A
C
C***    5 CALL TTIME(T(1))  !5 IS TRANSFERED TO THE NEXE LINE
        t(1)=second()

      READ (5,100,END=990) HED,NUMNP,NELTYP,LL,NF,NDYN,MODEX,NAD,
     1                     KEQB,N10SV
      IF(MODEX.GT.0) MODEX = 1
      IF (NUMNP.EQ.0) go to 1999
      WRITE (6,200) HED,NUMNP,NELTYP,LL,NF,NDYN,MODEX,NAD,KEQB,N10SV
      IF(KEQB.LT.2) KEQB = 99999
      IF (NDYN.NE.0) LL=1
      IF(LL.GE.1) GO TO 10
      WRITE (6,300)
      go to 1999
C***  DATA PORTHOLE SAVE
   10 IF(MODEX.EQ.1)
     *WRITE (NT8)      HED,NUMNP,NELTYP,LL,NF,NDYN
C
      KDYN = IABS(NDYN) +1
      IF(KDYN.LE.5) GO TO 14
      WRITE (6,310) NDYN
       go to 1999
C
C     RE-START MODE ACTIVATED IF  NDYN.EQ.-2  OR  NDYN.EQ.-3
C
   14 IF(NDYN.LT.0) GO TO 20
C
C     I N P U T   J O I N T   D A T A
C
      N2=N1+6*NUMNP
      N3=N2+NUMNP
      N4=N3+NUMNP
      N5=N4+NUMNP
      N6=N5+NUMNP
      IF(N6.GT.MTOT) CALL ERROR(N6-MTOT)
C
      CALL INPUTJ(A(N1),A(N2),A(N3),A(N4),A(N5),NUMNP,NEQ)
C
C     F O R M   E L E M E N T   S T I F F N E S S E S
C
C***       CALL TTIME(T(2))
         t(2)=second()
C
      MBAND=0
```

```
      NUMEL=0
      REWIND 1
      REWIND 2
C
      DO 900 M=1,NELTYP
      READ  (5,1001) NPAR
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1) WRITE (NT8) NPAR
      WRITE (1) NPAR
      NUMEL=NUMEL+NPAR(2)
      MTYPE=NPAR(1)
C
      CALL ELTYPE(MTYPE)
C
  900 CONTINUE
       neqq=neq
       numee=numel
C
C     D E T E R M I N E   B L O C K S I Z E
C
C     ADDSTF
C
      NEQB=(MTOT - 4*LL)/(MBAND + LL + 1)/2
C
C     OVER-RIDE THE SYSTEM MATRIX BLOCKSIZE WITH THE INPUT (NON-ZERO)
C     VALUE, KEQB.
C     THIS OVER-RIDE ENTRY IS TO ALLOW PROGRAM CHECKING OF MULTI-
C     BLOCK ALGORITHMS WITH WHAT WOULD NORMALLY BE ONE BLOCK DATA.
C
      IF(KEQB.LT.NEQB) NEQB = KEQB
C
      GO TO  (690,700,700,700,730), KDYN
C
C     STATIC SOLUTION
C
  690 CONTINUE
      NEQB1=(MTOT - MBAND)/(2*(MBAND+LL) + 1)
      NEQB2=(MTOT - MBAND - LL*(MBAND-2))/(3*LL + MBAND + 1)
      IF  (NEQB1.LT.NEQB) NEQB=NEQB1
      IF  (NEQB2.LT.NEQB) NEQB=NEQB2
      NBLOCK = (NEQ-1)/NEQB +1
      IF(NEQB.GT.NEQ) NEQB=NEQ
       neqb=neq
       nblock=1
      GO TO 790
C
C     EIGENSOLUTION
C
C        1.  DETERMINANT SEARCH ALGORITHM
C
  700 IF  (NEQB.LT.NEQ) GO TO 710
      NIM=3
      NC=NF + NIM
      NVM=6
      NCA=NEQ*MAXO(MBAND,NC)
```

```
      NTOT=NCA + 4*NEQ + 2*NVM*NEQ + 5*NC
      NEIG=0
      IF(NTOT.LE.MTOT) GO TO 720
C
C        2.  SUBSPACE ITERATION ALGORITHM
C
  710 NV=MINO(2*NF,NF+8)
      IF (NAD.NE.O) NV=NAD
      NEQB1=(MTOT - MBAND)/(2*MBAND + 1)
      NEQB2=(MTOT - MBAND - 2*NV - NV*(MBAND-2))/(3*NV + MBAND + 1)
      NEQB3=(MTOT - 3*NV*NV - 3*NV)/(2*NV + 1)
      NEQB4=(MTOT - 6*NV)/(1 + MBAND)
      IF (NEQB1.LT.NEQB) NEQB=NEQB1
      IF (NEQB2.LT.NEQB) NEQB=NEQB2
      IF (NEQB3.LT.NEQB) NEQB=NEQB3
      IF (NEQB4.LT.NEQB) NEQB=NEQB4
      NEIG=1
C
  720 CONTINUE
      NBLOCK = (NEQ-1)/NEQB +1
      IF (NEQB.GE.NEQ) NEQB=NEQ
C
C     HISTORY OR SPECTRUM ANALYSIS
C
      KREM = 1000
      NTOT = NBLOCK*NEQB*NF + KREM
      IF(MTOT.LT.NTOT)
     *WRITE (6,320)
      GO TO 790
C
C     STEP-BY-STEP DIRECT INTEGRATION
C
  730 CONTINUE
C     DISPLACEMENT COMPONENTS FOR DIRECT OUTPUT (*NSD*)
      NN2 = NEQ
C     DISPLACEMENT COMPONENTS REQUIRED FOR RECOVERY OF ALL OF THE
C     REQUESTED ELEMENT STRESS COMPONENTS (*NSS*)
      NN3 = NEQ
C
C        1.  DECOMPOSITION
C
      NEQB1 = (MTOT-NN2-NN3-NEQ-MBAND)/(2*MBAND+1)
C
C        2.  TIME INTEGRATION PHASE
C
      NEQB2 = (MTOT-MBAND-2*(NN2+NN3)-5*NEQ)/(MBAND+1)
C
      IF(NEQB1.LT.NEQB) NEQB = NEQB1
      IF(NEQB2.LT.NEQB) NEQB = NEQB2
      IF(NEQB.GT.NEQ)   NEQB = NEQ
      NBLOCK = (NEQ-1)/NEQB +1
C
C        3.  INPUT PHASE
C
C     NUMBER OF TIME FUNCTIONS (*NFN*)
```

```
      NN2 = 10
C     MAXIMUM NUMBER OF FUNCTION DEFINITION POINTS (*MXLP*)
      NN3 = 40
C
      NN4 = 6*NUMNP + 2*NN2*NEQ
      IF(NN4.GT.MTOT)
     *WRITE (6,320)
      NN4 = NEQ*2*(NN2+1) + NN2*(1+2*NN3)
      IF(NN4.GT.MTOT)
     *WRITE (6,320)
C
  790 CONTINUE
C
C     I N P U T   N O D A L   L O A D S
C
      N3=N2+NEQB*LL
      N4=N3+6*LL
      WRITE (6,201) NEQ,MBAND,NEQB,NBLOCK
C
C***      CALL TTIME(T(3))
          t(3)=second()


C
c         write(6,*)'# neqb,ll,n2,n3',neqb,ll,n2,n3
      CALL INL(A(N1),A(N2),A(N3),A(N4),NUMNP,NEQB,LL)
cl         do 16 l=n2,n3
c16       write(6,*)'# a(n2)',a(l)
C
C***      CALL TTIME(T(4))
         t(4)=second()
C
C     F O R M   T O T A L   S T I F F N E S S
C
      NE2B=2*NEQB
      N2=N1+NEQB*MBAND
      N3=N2+NEQB*LL
      N4=N3+4*LL
      NN2=N1+NE2B*MBAND
      NN3=NN2+NE2B*LL
      NN4=NN3+4*LL
        if(option.eq.1.)   call column
c         nn2=nl+nterms
c         nn3=nn2+neq*ll
c         nn4=nn3+4*ll
      ntr=nterms
C

      CALL ADDSTF (A(N1),A(NN2),A(NN3),A(NN4),NUMEL,NBLOCK,NE2B,LL,MBAND
     1,ANORM,NVV)
        if (option.eq.1.) then
        nl=1
        nm2=nl+nterms
        nnn3=nn2+neq*ll
        icount=nm2
        do 126 ii=nn2,nnn3-1
```

```
        a(icount)=a(ii)
        icount=icount+1
126     continue
        call assm(a(n1),a(nm2),11,nterms,neq)
         endif
c       write(6,*)'# nn2,nn3',nn2,nn3
c       do 17 l=n1,ntr
c17      write(6,*)'# a(ntr)',a(l)
C
C***      CALL TTIME(T(5))
            t(5)=second()
C
C     S O L U T I O N   P H A S E
C
          End barrier
   20 GO TO (30,40,50,60,70), KDYN
C
C     STATIC SOLUTION
C
   30 IF(MODEX.EQ.0) GO TO 32
      DO 31 I=6,10
   31 T(I) = T(5)
      GO TO 90
C
32        zzzx=0.
c   32 FORCECALL SOLEQ
           Forcecall SOLEQ
C***      CALL TTIME(T(6))
CCCCCCCVVVBBNM  the following barrier bkock is transfered fromm the end
       Barrier
      TT = 0.0
      DO 195 I=1,9
      T(I) = T(I+1)-T(I)
      TT = TT + T(I)
  195 CONTINUE
C
      WRITE (6,203) (T(K),K=1,9),TT
C
        End barrier
            Join
          Barrier
          t(6)=second()
      DO 33 I=7,10
   33 T(I) = T(6)
      GO TO 90
C
C     EIGENVALUE EXTRACTION
C
       End barrier
40       continue
           Barrier
      T(6) = T(5)
      CALL SOLEIG
C***      CALL TTIME(T(7))
        t(7)=second()
```

```
      T(8)  = T(7)
      T(9)  = T(7)
      T(10)= T(7)
      GO TO 90
C
C     FORCED DYNAMIC RESPONSE ANALYSIS
C
       End Barrier
50          continue
          Barrier
      T(6)  = T(5)
      IF(NDYN.LT.O) GO TO 52
      CALL SOLEIG
C***        CALL TTIME  (T(7))
               t(7)=second()
      GO TO 54
   52 DO 53 I=1,6
   53 T(I+1)=T(I)
      REWIND 2
      READ  (2)  NEQ,NBLOCK,NEQB,MBAND,N1,NF,(QQQ(I),I=1,NF)
      REWIND 7
      IMAX=NEQB*NF
      READ  (7)  (A(I),I=1,NF)
      DO 56 L=1,NBLOCK
   56 READ  (7)  (A(I),I=1,IMAX)
   54 CALL HISTRY
C***        CALL TTIME  (T(8))
         t(8)=second()
      T(9)  = T(8)
      T(10)= T(8)
      GO TO 90
C
C     RESPONSE SPECTRUM ANALYSIS
C
      End barrier
  60   continue
        Barrier
      T(6)  = T(5)
      IF(NDYN.LT.O) GO TO 62
      CALL SOLEIG
         t(7)=second()
C***        CALL TTIME  (T(7))
      T(8)  = T(7)
      GO TO 64
   62 DO 63 I=1,7
   63 T(I+1)=T(I)
      REWIND 2
      READ  (2)  NEQ,NBLOCK,NEQB,MBAND,N1,NF
      REWIND 7
      IMAX=NEQB*NF
      READ  (7)  (A(I),I=1,NF)
      DO 66 L=1,NBLOCK
   66 READ  (7)  (A(I),I=1,IMAX)
   64 CALL RESPEC
C***        CALL TTIME  (T(9))
```

```
         t(9)=second()
      T(10)= T(9)
      GO TO 90
C
C     STEP-BY-STEP (DIRECT INTEGRATION) ANALYSIS
C
       End barrier
   70    continue
        Barrier
      DO 71 I=6,9
   71 T(I) = T(5)
      CALL STEP
C***       CALL TTIME(T(10))
         t(10)=second()
C
C     COMPUTE AND PRINT OVERALL TIME LOG
C
       End barrier
90    continue
       Barrier
      TT = 0.0
      DO 95 I=1,9
      T(I) = T(I+1)-T(I)
      TT = TT + T(I)
   95 CONTINUE
C
      WRITE (6,203) (T(K),K=1,9),TT
C
        End barrier
      GO TO 5
c  990 continue
c1999  continue
C
  100 FORMAT (12A6/9I5)
  200 FORMAT(1H1,12A6///
     1  38H C O N T R O L   I N F O R M A T I O N, // 4X,
     2  27H NUMBER OF NODAL POINTS    =, I5 / 4X,
     3  27H NUMBER OF ELEMENT TYPES   =, I5 / 4X,
     4  27H NUMBER OF LOAD CASES      =, I5 / 4X,
     5  27H NUMBER OF FREQUENCIES     =, I5 / 4X,
     6  27H ANALYSIS CODE (NDYN)      =, I5 / 4X,
     7  16H   EQ.0,  STATIC,               / 4X,
     8  26H   EQ.1,  MODAL EXTRACTION,     / 4X,
     9  25H   EQ.2,  FORCED RESPONSE,      / 4X,
     A  27H   EQ.3,  RESPONSE SPECTRUM,    / 4X,
     *  28H   EQ.4,  DIRECT INTEGRATION,   / 4X,
     B  27H SOLUTION MODE (MODEX)     =, I5 / 4X,
     C  19H   EQ.0,  EXECUTION,            / 4X,
     D  20H   EQ.1,  DATA CHECK,           / 4X,
     E  19H NUMBER OF SUBSPACE,            / 4X,
     F  27H ITERATION VECTORS (NAD)  =, I5 / 4X,
     G  27H EQUATIONS PER BLOCK       =, I5 / 4X,
     H  27H TAPE10 SAVE FLAG (N10SV) =, I5 / 4X)
  201 FORMAT (38H1E Q U A T I O N   P A R A M E T E R S, //
     *        34H  TOTAL NUMBER OF EQUATIONS       =,I5,
```

```
     1      /34H  BANDWIDTH                        =,I5,
     2      /34H  NUMBER OF EQUATIONS IN A BLOCK =,I5,
     3      /34H  NUMBER OF BLOCKS                 =,I5)
  203 FORMAT (1H1,31H0 V E R A L L   T I M E   L O G, //
     1 5X,30HNODAL POINT INPUT             =, F8.2 /
     2 5X,30HELEMENT STIFFNESS FORMATION   =, F8.2 /
     3 5X,30HNODAL LOAD INPUT              =, F8.2 /
     4 5X,30HTOTAL STIFFNESS FORMATION     =, F8.2 /
     5 5X,30HSTATIC ANALYSIS               =, F8.2 /
     6 5X,30HEIGENVALUE EXTRACTION         =, F8.2 /
     7 5X,30HFORCED RESPONSE ANALYSIS      =, F8.2 /
     8 5X,30HRESPONSE SPECTRUM ANALYSIS    =, F8.2 /
     * 5X,30HSTEP-BY-STEP INTEGRATION      =, F8.2 //
     9 5X,30HTOTAL SOLUTION TIME           =, F8.2 /)
C
  300 FORMAT (// 48H ** ERROR.  (AT LEAST ONE LOAD CASE IS REQUIRED)  )
  310 FORMAT (// 33H ** ERROR.  ANALYSIS CODE (NDYN =,I3,9H) IS BAD.  )
  320 FORMAT (// 47H ** WARNING.  ESTIMATE OF STORAGE FOR A DYNAMIC,
     1           32H ANALYSIS EXCEEDS AVAILABLE CORE, // 1X)
C
 1001 FORMAT (14I5)
c        End barrier
990      continue
1999      continue
c      Join
      END
      SUBROUTINE CALBAN (MBAND,NDIF,LM,XM,S,P,ND,NDM,NS)
c      IMPLICIT REAL*8(A-H,O-Z)
C
C     CALLED BY?  RUSS,TEAM,PLNAX,BRICK8,TPLATE,CLAMP,ELST3D,PIPEK
C
C-----CALCULATES BAND WIDTH AND WRITES STIFFNESS MATRIX ON TAPE 2
      DIMENSION LM(1),XM(1),S(NDM,NDM),P(NDM,4)
      COMMON /EXTRA/ MODEX,NT8,IFILL(14)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
c      write(6,*)' sub calban starts'
      neq=neqq
      nume=numee
      MIN=100000
      MAX=0
      DO 800 L=1,ND
      IF (LM(L).EQ.0) GO TO 800
      IF (LM(L).GT.MAX) MAX=LM(L)
      IF (LM(L).LT.MIN) MIN=LM(L)
  800 CONTINUE
      NDIF=MAX-MIN+1
      IF (NDIF.GT.MBAND) MBAND=NDIF
      IF(MODEX.EQ.1) GO TO 810
C
      LRD=ND*(ND+1)/2+5*ND
      WRITE(2) LRD,ND,(LM(I),I=1,ND),((S(I,J),J=1,ND),I=1,ND),
     1 ((P(I,J),I=1,ND),J=1,4),(XM(I),I=1,ND)
        write(14)lrd,nd,(lm(i),i=1,nd)
c       rewind 13
```

```
        write(13) ((s(i,j),j=1,nd),i=1,nd)
c      moayyad
c       write(6,*)' sub. calban.......'
c       write(6,*)'lrd,nd,(lm(i),i=1,nd),((s(i,j),j=1(=i),nd),i=1,nd)'
c       write(6,*)'((p(i,j),i=1,nd),j=1,4),(xm(i),i=1,nd)'
c        write(6,*)'  lrd     nd',lrd,nd
c        write(6,*)'==========s=======**'
c        write(6,115) ((s(i,j),j=1,nd),i=1,nd)
c        write(6,*)'==========p======='
c        write(6,115) ((p(i,j),i=1,nd),j=1,4)
c        write(6,*)'==========xm=========*'
c        write(6,115) (xm(i),i=1,nd)
c        write(6,*)'=================='
115      format(6e12.5)
c        write(6,*)'sub calban ends'
*****************************************************************VVVVVVVVVV
c      initialize all row length (include the diagonal)
c       do 1 i=1,neq
c1      irowl(i)=0
c       do 2 i=1,nume
       maxdof=0
       do 3 j1=1,nd
       jj1=lm(j1)
       if(jj1.gt.maxdof) maxdof=jj1
3      continue
c      find the current row length and update the row length
       do 4 j1=1,nd
       jj1=lm(j1)
       if (jj1.eq.0) go to 4
       nowr1=maxdof-jj1+1
       if(nowr1.gt.irowl(jj1))irowl(jj1)=nowr1
c       write(6,*)' jj1 irowl nd nume...calb',jj1,irowl(jj1),nd,nume
4      continue
c2      continue
cnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
      RETURN
C
  810 WRITE (1) ND,NS,(LM(I),I=1,ND)
      RETURN
      END
*****************************************************************************
      SUBROUTINE ELTYPE(MTYPE)
C
c       IMPLICIT REAL*8(A-H,O-Z)
C
C     CALLED BY?  MAIN,STRESS
C
c      common /maybe/ dxx(50),dyy(50),dzz(50),ee(50),aa(50)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      GO TO (1,2,3,4,5,6,7,8,9,10,11,12),MTYPE
C
C     THREE DIMENSIONAL TRUSS ELEMENTS
C
c       write(6,*)' sub eltype begins'
```

```
    1 CALL TRUSS
      GO TO 900
C
C     THREE DIMENSIONAL BEAM ELEMENTS
C
    2 CALL BEAM
      GO TO 900
C
C     PLANE STRESS ELEMENTS
C
    3 CALL PLANE
      GO TO 900
C
C     AXISYMMETRIC SOLID ELEMENTS
C
    4 CALL PLANE
      GO TO 900
C
C     THREE DIMENSIONAL SOLID ELEMENTS
C
    5 CALL THREED
      GO TO 900
C
C     PLATE BENDING ELEMENTS
C
    6 CALL SHELL
      GO TO 900
C
C
    7 CALL BOUND
      GO TO 900
C
C     THICK SHELL ELEMENTS
C
    8 CALL SOL21
      GO TO 900
C
    9 WRITE (6,100) MTYPE
      GO TO 900
C
   10 WRITE (6,100) MTYPE
      GO TO 900
C
   11 WRITE (6,100) MTYPE
      GO TO 900
C
C     STRAIGHT OR CURVED PIPE ELEMENTS
C
   12 CALL PIPE
C
c900       write(6,*)' sub. eltype ends'
900   RETURN
C
  100 FORMAT ('0ELEMENT',I4,' IS NOT IMPLEMENTED YET')
      END
```

```
************************************************************************
      SUBROUTINE INL(ID,B,TR,TMASS,NUMNP,NEQB,LL)
C
c      IMPLICIT REAL*8(A-H,O-Z)
C
C     CALLED BY?  MAIN
C
C     INPUT NODAL LOADS AND MASSES
C
      DIMENSION ID(NUMNP,6),B(NEQB,LL),TR(6,LL),TMASS(NEQB)
      COMMON / JUNK / R(6),TXM(6),IFILL1(406)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
c       write(6,*)' sub inl begins'
      NT=3
      REWIND NT
      KSHF=0
      WRITE (6,2002)
      IF(MODEX.EQ.1) GO TO 50
      DO 750 I=1,NEQB
      TMASS(I)=0.
      DO 750 K=1,LL
  750 B(I,K)=0.0
C
   50 DO 900 NN=1,NUMNP
C
      DO 100 I=1,6
      TXM(I)=0.
      DO 100 J=1,LL
  100 TR(I,J)=0.0
C
      IF(NN.EQ.1) GO TO 300
  150 IF(N.NE.NN) GO TO 400
      DO 200 I=1,6
      IF  (L) 180,180,190
  180 TXM(I)=R(I)
      GO TO 200
  190 TR(I,L)=R(I)
  200 CONTINUE
  300 READ (5,1001) N,L,R
      IF  (N.EQ.0) GO TO 150
      WRITE(6,2001) N,L,R
      GO TO 150
C
  400 IF(MODEX.EQ.1) GO TO 900
      DO 800 J=1,6
      II=ID(NN,J)-KSHF
      IF  (II) 800,800,500
  500 DO 600  K=1,LL
  600 B(II,K)=TR(J,K)
      TMASS(II)=TXM(J)
  610 IF(II.NE.NEQB) GO TO 800
c      write(6,*)' nt',nt
      WRITE (NT) B,TMASS
c       rewind 13
```

```
c       write(13) b,tmass
c       do 29 n=1,neqb
c29     write(6,*)' load b',(b(n,m),m=1,11)
      KSHF=KSHF+NEQB
      DO 700 I=1,NEQB
      TMASS(I)=0.
      DO 700 K=1,LL
  700 B(I,K)=0.0
  800 CONTINUE
  900 CONTINUE
C
      IF(MODEX.EQ.1) RETURN
C
      WRITE (NT) B,TMASS
c       write(13)b,tmass
c       do 19 i=1,neqb
c19     write(6,*)' load b',(b(i,j),j=1,11)
C
c      write(6,*)' sub inl ends'
      RETURN
 1001 FORMAT (2I5,7F10.4)
 2001 FORMAT (2(3X,I4),6E15.5)
 2002 FORMAT (47H1N O D A L   L O A D S   (S T A T I C)   O R   ,
     A        29HM A S S E S   (D Y N A M I C), ///
     B        3X,4HNODE,3X,4HLOAD,
     1 2(9X,6HX-AXIS,9X,6HY-AXIS,9X,6HZ-AXIS), / 7H NUMBER,3X,4HCASE,
     2 3(10X,5HFORCE), 3(9X,6HMOMENT), / 1X)
      END
**********************************************************************
      SUBROUTINE INPUTJ(ID,X,Y,Z,T,NUMNP,NEQ)
C
c      IMPLICIT REAL*8(A-H,O-Z)
C
C     CALLED BY?  MAIN
C
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,6),T(1)
C     .
      COMMON /EXTRA/ MODEX,NT8,IFILL(14)
C
C---- SPECIAL NODE CARD FLAGS
C
C     IT    =   COORDINATE SYSTEM TYPE  (CC 1, ANY NODE CARD)
C               EQ.C, CYLINDRICAL
C     IPR   =   PRINT SUPPRESSION FLAG  (CC 6, CARD FOR NODE 1 ONLY)
C               EQ. , NORMAL PRINTING
C               EQ.A, SUPPRESS SECOND PRINTING OF NODAL ARRAY DATA
C               EQ.B, SUPPRESS PRINTING OF ID-ARRAY
C               EQ.C, BOTH *A* AND *B*
C
      DIMENSION IPRC(4)
C
      DATA IPRC/1H ,1HA,1HB,1HC/
C
c        write(6,*)' sub. inputj begins....'
      IPR = IPRC(1)
```

```
      RAD = ATAN(1.0D0)/45.0D0
C
C
C---- READ OR  GENERATE NODAL POINT DATA--------------------------------
      WRITE (6,2000)
      WRITE (6,2001)
      NOLD=0
   10 READ  (5,1000) IT,N,JPR,(ID(N,I),I=1,6),X(N),Y(N),Z(N),KN,T(N)
      WRITE (6,2002) IT,N,JPR,(ID(N,I),I=1,6),X(N),Y(N),Z(N),KN,T(N)
      IF(N.EQ.1) IPR = JPR
      IF(IT.NE.IPRC(4)) GO TO 15
      DUM = Z(N)* RAD
      Z(N)  = X(N)*COS(DUM)
      X(N)  = X(N)*SIN(DUM)
   15 CONTINUE
      IF(NOLD.EQ.0) GO TO 50
C-----CHECK IF GENERATION IS REQUIRED-----------------------------------
      DO 20 I=1,6
      IF(ID(N,I).EQ.0.AND.ID(NOLD,I).LT.0) ID(N,I)=ID(NOLD,I)
   20 CONTINUE
      IF(KN.EQ.0) GO TO 50
      NUM=(N-NOLD)/KN
      NUMN=NUM-1
      IF(NUMN.LT.1) GO TO 50
      XNUM=NUM
      DX=(X(N)-X(NOLD))/XNUM
      DY=(Y(N)-Y(NOLD))/XNUM
      DZ=(Z(N)-Z(NOLD))/XNUM
      DT=(T(N)-T(NOLD))/XNUM
      K=NOLD
      DO 30 J=1,NUMN
      KK=K
      K=K+KN
      X(K)=X(KK)+DX
      Y(K)=Y(KK)+DY
      Z(K)=Z(KK)+DZ
      T(K)=T(KK)+DT
      DO 30 I=1,6
      ID(K,I)=ID(KK,I)
      IF (ID(K,I).GT.1) ID(K,I)=ID(KK,I)+KN
   30 CONTINUE
C
   50 NOLD=N
      IF(N.NE.NUMNP) GO TO 10
C
C---- PRINT ALL NODAL POINT DATA----------------------------------------
C
      IF(IPR.EQ.IPRC(2) .OR. IPR.EQ.IPRC(4)) GO TO 52
      WRITE (6,2003)
      WRITE (6,2001)
      WRITE (6,2005) (N,(ID(N,I),I=1,6),X(N),Y(N),Z(N),T(N),N=1,NUMNP)
   52 CONTINUE
C
C-----NUMBER UNKNOWNS AND SET MASTER NODES NEGATIVE---------------------
C
```

```
      NEQ=0
      DO 60 N=1,NUMNP
      DO 60 I=1,6
      ID(N,I)=IABS(ID(N,I))
      IF(ID(N,I)-1) 57,58,59
   57 NEQ=NEQ+1
      ID(N,I)=NEQ
      GO TO 60
   58 ID(N,I)=0
      GO TO 60
   59 ID(N,I)=-ID(N,I)
   60 CONTINUE
C
C---- PRINT MASTER INDEX ARRAY
C
      IF(IPR.EQ.IPRC(3) .OR. IPR.EQ.IPRC(4)) GO TO 62
      WRITE (6,2004) (N,(ID(N,I),I=1,6),N=1,NUMNP)
   62 CONTINUE
      IF(MODEX.EQ.0) GO TO 70
C***  DATA PORTHOLE SAVE
      WRITE (NT8) ((ID(N,I),I=1,6),N=1,NUMNP)
      WRITE (NT8) (X(N),N=1,NUMNP)
      WRITE (NT8) (Y(N),N=1,NUMNP)
      WRITE (NT8) (Z(N),N=1,NUMNP)
      WRITE (NT8) (T(N),N=1,NUMNP)
      ENDFILE NT8
C
      REWIND 2
      WRITE (2) ID
C
      RETURN
C
   70 CONTINUE
      REWIND 8
      WRITE (8) ID
C
      RETURN
C
 1000 FORMAT (2(A1,I4),5I5,3F10.0,I5,F10.0)
 2000 FORMAT (//23H NODAL POINT INPUT DATA )
 2001 FORMAT (5HONODE 3X 24HBOUNDARY CONDITION CODES 11X,
     .  23HNODAL POINT COORDINATES / 7H NUMBER 2X 1HX 4X 1HY 4X 1HZ 3X,
     . 2HXX 3X 2HYY 3X 2HZZ12X 1HX 12X 1HY 12X 1HZ 12X 1HT )
 2002 FORMAT (1X,A1,I4,A1,I3,5I5,3F13.3,I5,F13.3)
 2003 FORMAT (//21H1GENERATED NODAL DATA)
 2004 FORMAT (//17H1EQUATION NUMBERS/
     1 35H    N    X    Y    Z   XX   YY   ZZ /(7I5))
 2005 FORMAT (I5,6I5,4F13.3)
      END
*****************************************************************
      SUBROUTINE RUSS (ID,X,Y,Z,T,E,THERM,DEN,AREA,WT,NUMNP)
c       IMPLICIT REAL*8(A-H,O-Z)
C
C     CALLS?  CALBAN
C     CALLED BY?  TRUSS
```

```
C
C
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,1),E(1),THERM(1),DEN(1),AREA(1)
     . ,T(1),WT(1)
      COMMON /ELPAR/ NPAR(14),NNNNN,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/LM(24),ND,NS,S(24,24),P(24,4),XM(24),ST(12,24),TT(12,4)
     1           ,IFILL2(3048)
      COMMON /JUNK/ EMUL(4,4),I,J,K,L,M,N,II,JJ,KK,MTYPE,TEMP,DX,DY,DZ,
     1 XL2,XL,XX,YY,F,FT,FX,FY,FZ,MIN,MAX,NDIF,KKK,TEM,MTYP,IFILL1(355)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
c       common /maybe/ dxx(50),dyy(50),dzz(50),ee(50),aa(50)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
C     CONTROL INFORMATION AND MEMBER PROPERTIES
C
c      write(6,*)' sub russ begins'
      NUME=NPAR(2)
      NUMMAT=NPAR(3)
      neqq=neq
      numee=nume
      WRITE (6,2000) NUME,NUMMAT
      WRITE (6,2001)
      DO 10 I=1,NUMMAT
      READ  (5,1001) N,E(N),THERM(N),DEN(N),AREA(N),WT(N)
   10 WRITE (6,2002) N,E(N),THERM(N),DEN(N),AREA(N),WT(N)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) (E(N),THERM(N),DEN(N),AREA(N),WT(N),N=1,NUMMAT)
C
C     ELEMENT LOAD MULTIPLIERS
C
      READ (5,1003)  EMUL
      WRITE (6,2003) EMUL
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) EMUL
C
C     ELEMENT INFORMATION
      WRITE (6,2005)
C
      N=1
  100 READ (5,1004)  M,II,JJ,MTYP,TEM,KK
      IF(KK.EQ.0) KK=1
  120 IF(M.NE.N) GO TO 200
      I=II
      J=JJ
      MTYPE=MTYP
      REFT=TEM
      KKK=KK
C
C     1. FORM ELEMENT STIFFNESS AND STRESS MATRICES
C
  200 CONTINUE
      IF(MODEX.EQ.1) GO TO 380
```

```
      DX=X(I)-X(J)
      DY=Y(I)-Y(J)
      DZ=Z(I)-Z(J)
c      dxx(m)=dx
c      dyy(m)=dy
c       dzz(m)=dz
      XL2=DX*DX+DY*DY+DZ*DZ
      XL=SQRT(XL2)
      XX=E(MTYPE)*AREA(MTYPE)*XL
c      ee(m)=e(mtype)
c      aa(m)=area(mtype)
      ST(1,1)=DX/XL2
      ST(1,2)=DY/XL2
      ST(1,3)=DZ/XL2
      ST(1,4)=-ST(1,1)
      ST(1,5)=-ST(1,2)
      ST(1,6)=-ST(1,3)
C
      DO 300 L=1,6
      YY=ST(1,L)*XX
      DO 250 K=L,6
      S(K,L)=ST(1,K)*YY
  250 S(L,K)=S(K,L)
      ST(1,L)=E(MTYPE)*ST(1,L)
  300 ST(2,L)=AREA(MTYPE)*ST(1,L)
C
C     2. INERTIA AND THERMAL LOADS
C
      F=WT(MTYPE)*AREA(MTYPE)*XL/2.
      TEMP=(T(I)+T(J))*0.5 - REFT
      FT=TEMP*THERM(MTYPE)*E(MTYPE)*AREA(MTYPE)
      FT = -FT
      FX=DX*FT/XL
      FY=DY*FT/XL
      FZ=DZ*FT/XL
C
      DO 350 L=1,4
      TT(2,L)=EMUL(L,4)*FT
      TT(1,L)=TT(2,L)/AREA(MTYPE)
      P(1,L)=EMUL(L,1)*F-EMUL(L,4)*FX
      P(2,L)=EMUL(L,2)*F-EMUL(L,4)*FY
      P(3,L)=EMUL(L,3)*F-EMUL(L,4)*FZ
      P(4,L)=EMUL(L,1)*F+EMUL(L,4)*FX
      P(5,L)=EMUL(L,2)*F+EMUL(L,4)*FY
  350 P(6,L)=EMUL(L,3)*F+EMUL(L,4)*FZ
      F=DEN(MTYPE)*AREA(MTYPE)*XL/2.
      DO 375 L=1,6
  375 XM(L)=F
  380 CONTINUE
C
C     3. FORM LOCATION MATRIX AND COMPUTE BAND WIDTH
C
      DO 400 L=1,3
      LM(L)=ID(I,L)
  400 LM(L+3)=ID(J,L)
```

```
C
      ND=6
      NS=2
      NDM=24
      CALL CALBAN (MBAND,NDIF,LM,XM,S,P,ND,NDM,NS)
      IF(MODEX.EQ.0) GO TO 410
C***  DATA PORTHOLE SAVE
      WRITE (NT8) N,I,J,MTYPE,REFT
      GO TO 420
  410 CONTINUE
      WRITE (1) ND,NS,(LM(L),L=1,ND),((ST(L,K),L=1,NS),K=1,ND),
     1 ((TT(L,K),L=1,NS),K=1,4)
c        write(6,*)'% nd,ns',nd,ns
c         do 88 l=1,ns
c88        write(6,87)' st',(st(l,k),k=1,nd)
c87        format(6f10.1)
C
C     4. CHECK FOR MORE ELEMENTS
C
  420 CONTINUE
      WRITE (6,2004) N,I,J,MTYPE,REFT,NDIF
      IF (N.EQ.NUME) RETURN
      N=N+1
      I=I+KKK
      J=J+KKK
      IF(N.GT.M) GO TO 100
      GO TO 120
C
 1001 FORMAT (I5,5F10.0)
 1003 FORMAT (4F10.0)
 1004 FORMAT (4I5,1F10.0,I5)
 2000 FORMAT (///25H1NUMBER OF TRUSS MEMBERS= I5/
     1 25H NUMBER OF DIFF. MEMBERS= I5)
 2001 FORMAT (///1X,4HTYPE,14X,1HE,10X,5HALPHA,12X,3HDEN,11X,4HAREA
     1 11X,4H WT )
 2002 FORMAT (I5,5E15.7)
 2003 FORMAT(///25H ELEMENT LOAD MULTIPLIERS / 20X,1HA,14X,1HB,14X,1HC,
     1 14X,1HD,/6H X-DIR4E15.6/ 6H Y-DIR4E15.6/ 6H Z-DIR4E15.6/
     2 6H  TEMP4E15.6)
 2004 FORMAT (4I6,F10.2,I7)
 2005 FORMAT (///42H1    N     I     J  TYPE      TEMP   BAND  )
      END
*******************************************************************
              Forcesub SOLEQ of NNP ident ME
c       SUBROUTINE SOLEQ
c       IMPLICIT REAL*8(A-H,O-Z)
C
C     CALLS?  SESOL,PRINTD,STRESS
C     CALLED BY?  MAIN
C
C     STATIC SOLUTION PHASE
C
       COMMON /one/A(1)
      COMMON /ELPAR/ NP(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /SOL  / NBLOCK,NEQB,LL,NF,IFILL(7)
```

```
       common /say/ neqq,numee,loopur,nnblock,nterms,option
       common /what/ naxa(10000),irowl(10000),icolh(10000)
       common /time/ t1(8),t2(8),t3(8)
c             dimension bb(100),b(3,1)
       integer iops(8),iopf(8)
C
      REAL TT(4)
        End declarations
C
C     SOLVE FOR THE DISPLACEMENT VECTORS
C
C***       CALL TTIME(TT(1))
          if(me.eq.1)  tt(1)=second()
c          write(6,*)  ' sub soleq begins'
            Barrier
      NSB=(MBAND+LL)*NEQB
      NSBB=NEQB*LL*(2+(MBAND-2)/NEQB)
       IF(NSBB.LT.NSB)  NSBB=NSB
      N4=N3+NSBB
      MI = MBAND + NEQB -1
c      moayyad
          if  (option.eq.1.)  then
        do 119  i=1,neqq
119     irowl(i)=irowl(i)-1
          n1=1
          n2=n1+nterms
c        call xload(neqq,11,a(n2))
c         do 198  i1=1,nterms
c198      write(6,*)' a vector before row9',a(i1)
c         do 199  i1=n2,neqq
c199      write(6,*)' load vector before row9',a(i1)
           endif
          End barrier
             if(option.eq.1.)  then
                 neqq=neq
          neqp1=neq+1
          if(me.eq.1)ts1=second()
        t1(me)=second()
        Forcecall row9(a(n1),a(n2),naxa,irowl,icolh,neqq,neqp1,nterms,
     &1,iopf(me),11)
c        write(6,*)' factorization ends....'
              t2(me)=second()
        write(16,*)' Factorization time of proc.',me,'is',t2(me)-t1(me)
        Forcecall row9(a(n1),a(n2),naxa,irowl,icolh,neqq,neqp1,nterms,
     &2,iops(me),11)
             t2(me)=second()
        write(16,*)' Eqn solver time of proc.',me,'is',t2(me)-t1(me)
         if(me.eq.1)then
         ts2=second()
         tst=ts2-ts1
         write(16,*)' cpu time for the eqn solver:',tst
         endif
           else
            Barrier
       CALL SESOL  (A(N1),A(N3),A(N4),NEQ,MBAND,LL,NBLOCK,NEQB,NSB,MI,
```

```
     1              4,3,2,7)
           End barrier
             endif
C***       CALL TTIME(TT(2))
          if(me.eq.1)tt(2)=second()
C
C      PRINT DISPLACEMENTS
C
         Barrier
       N2=N1+NUMNP*6
       N3=N2+6*LL
          if(option.eq.1.) then
          nblock=1

            neqb=neq
            Endif
       CALL PRINTD (A(N1),A(N2),A(N3),NEQB,NUMNP,LL,NBLOCK,NEQ,2,1)
C***       CALL TTIME(TT(3))
            tt(3)=second()
C
C      COMPUTE AND PRINT ELEMENT STRESSES
C
       N2=N1+4*LL
       N3=N2+NEQB*LL
       LB=(MTOT-N3)/(NEQ +12)
       CALL STRESS(A(N1),A(N2),A(N3),NEQB,LB,LL,NEQ,NBLOCK)
C***       CALL TTIME(TT(4))
          tt(4)=second()
C
C      COMPUTE TIME LOG FOR THE STATIC SOLUTION PHASE
C
       DO 50 K=1,3
   50 TT(K) = TT(K+1)-TT(K)
       WRITE (6,2000) (TT(L),L=1,3)
C
 2000 FORMAT (//// 48H S T A T I C   S O L U T I O N   T I M E   L O G,
     1        //5X,21HEQUATION SOLUTION    =, F8.2 /
     2          5X,21HDISPLACEMENT OUTPUT =, F8.2 /
     3          5X,21HSTRESS RECOVERY      =, F8.2 /)
C
c        write(6,*)' sub soleq ends'
         End barrier
       RETURN
       END
*************************************************************
       SUBROUTINE STRESS(STR,B,D,NEQB,LB,LL,NEQ,NBLOCK)
c        IMPLICIT REAL*8(A-H,O-Z)
C
C      CALLS?  ELTYPE
C      CALLED BY?  SOLEQ
C
       DIMENSION D(NEQ,LB),B(NEQB,LL),STR(4,LL)
       COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,MEQ
       COMMON /JUNK/ LT,LH,IFILL(428)
       COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
```

```
c       write(6,*)' sub stress begins  '
C
      READ (8) STR
      NT=(LL-1)/LB +1
      LH=0
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) NELTYP,NT
C
      DO 1000 II=1,NT
C
      LT =LH+1
      LLT=1-LT
      LH=LT+LB-1
      IF(LH.GT.LL) LH=LL
C
C     MOVE DISPLACEMENTS INTO CORE FOR LB LOAD CONDITIONS
C
      REWIND 2
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) LT,LH
      NQ=NEQB*NBLOCK
      DO 200 NN=1,NBLOCK
      READ (2) B
      N=NEQB
      IF (NN.EQ.1) N=NEQ-NQ+NEQB
      NQ=NQ-NEQB
      DO 200 J=1,N
      I=NQ+J
      DO 200 L=LT,LH
      K=L+LLT
  200 D(I,K)=B(J,L)
      LK=LH-LT+1
C
C     CALCULATE STRESSES FOR ALL ELEMENTS FOR LB LOAD CONDITIONS
C
      REWIND 1
      DO 1000 M=1,NELTYP
      READ (1) NPAR
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) NPAR
      MTYPE=NPAR(1)
      NPAR(1)=0
      CALL ELTYPE(MTYPE)
 1000 CONTINUE
C
c       write(6,*)' sub stress ends'
      RETURN
      END
************************************************************************
      SUBROUTINE STRSC(STR,D,NEQ,NTAG)
c      IMPLICIT REAL*8(A-H,O-Z)
C
```

```
C       CALLED BY?  TRUSS,BEAM,PLANE,THREED,SHELL,BOUND,PIPE
C
        DIMENSION STR(4,1),D(NEQ,1)
        COMMON /JUNK/ LT,LH,L,IPAD,SG(20),SIG(7),EXTRA(186)
        COMMON /EM/ NS,ND,B(42,63),TI(42,4),LM(63)
C
c        write(6,*)' sub strsc bigins'
        IF (NTAG.EQ.0) GO TO 800
        LL=L-LT+1
        DO 300 I=1,NS
        SG(I)=0.0
        DO 300 J=1,4
  300   SG(I)=SG(I)+TI(I,J)*STR(J,L)
        DO 500 J=1,ND
        JJ=LM(J)
        IF(JJ.EQ.0) GO TO 500
        DO 400 I=1,NS
  400   SG(I)=SG(I)+B(I,J)*D(JJ,LL)
C
  500   CONTINUE
        GO TO 900
  800   READ  (1) ND,NS,(LM(I),I=1,ND),(( B(I,J),I=1,NS),J=1,ND),
       1 ((TI(I,J),I=1,NS),J=1,4)
  900   RETURN
        END
*************************************************************************
        SUBROUTINE TRUSS
C
c        IMPLICIT REAL*8(A-H,O-Z)
C       CALLS? RUSS,STRSC
C       CALLED BY?  ELTYPE
C
        COMMON /one/A(1)
        COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
        COMMON /JUNK/  LT,LH,L,IPAD,SIG(20),N6,N7,N8,N9,N10,IFILL(381)
        COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
c         common /maybe/ dxx(50),dyy(50),dzz(50),ee(50),aa(50)
         common /say/ neqq,numee,loopur,nnblock,nterms,option
        common /what/ naxa(10000),irowl(10000),icolh(10000)
C
c         write(6,*)' sub truss begins'
        IF(NPAR(1).EQ.0) GO TO 500
        N6=N5+NUMNP
        N7 =N6+NPAR(3)
        N8 =N7+NPAR(3)
        N9 =N8+NPAR(3)
        N10=N9+NPAR(3)
        MM=N10+NPAR(3)-MTOT
        IF(MM.GT.0) CALL ERROR(MM)
C
        CALL RUSS(A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),A(N9),
       1              A(N10),NUMNP)
C
        RETURN
C
```

```
  500 WRITE (6,2002)
      NUME=NPAR(2)
      DO 800 MM=1,NUME
      CALL STRSC (A(N1),A(N3),NEQ,0)
      WRITE (6,2001)
      DO 800 L=LT,LH
      CALL STRSC (A(N1),A(N3),NEQ,1)
      WRITE (6,3002) MM,L,SIG(1),SIG(2)
C***  STRESS PORTHOLE
      IF(N1OSV.EQ.1)
     *WRITE (NT10) MM,L,SIG(1),SIG(2)
  800 CONTINUE
      RETURN
C
 2001 FORMAT (/)
 2002 FORMAT (//23H1  TRUSS MEMBER ACTIONS //
     .          46H0 MEMBER    LOAD          STRESS            FORCE   )
 3002 FORMAT (2I8,F15.5,F15.3)
      END
***********************************************************************
         subroutine printd (id,d,b,neqb,numnp,11,nblock,neq,nt,mq)
c         implicit real*8(a-h,o-z)
c
c        called by: soleq,soleig,respec
c
         dimension id(numnp,6),b(neqb,11),d(6,11)
       data q11,q21,q12,q22,q13,q23/' load',' case','eigen-','vector',
     &         ' mode ','number'/
c
c          write(6,*)' sub printd begind'
         rewind 8
         read(8) id
         m=neq
         nn=neqb*nblock
c
         if(mq.eq.2) go to 50
         if(mq.eq.3) go to 55
         rewind nt
         q1=q11
         q2=q21
         go to 60
50       q1=q12
         q2=q22
         go to 60
55       q1=q13
         q2=q23
         rewind nt
         read(nt)
60       write(6,2003) q1,q2
         n=numnp
c
         do 500 kk=1,numnp
c
         i=6
         do 250 ii=1,6
```

```
         do 100 l=1,11
100      d(i,l)=0.
         if(m.gt.nn) go to 150
         if(m.eq.0) go to 150
         read(nt) b
         nn=nn-neqb
150      if(id(n,i).lt.1) go to 250
         k=m-nn
         m=m-1
c
         do 200 l=1,11
200      d(i,l)=b(k,l)
250      i=i-1
c
         write(6,2004)n,(l,(d(i,l),i=1,6),l=1,11)
c
500      n=n-1
c
c         write(6,*)' sub printd ends'
         return
c
2003     format(1h1,' n o d e   d i s p l a c e m e n t s ',/,
     &   ' r o t a t i o n s ',// 3x,4hnode,2x,a6,2(12x,2hx-,12x,
     &    2hy-,12x,2hz-),/7h number ,2x,a6,3(3x,11htranslation ),
     &    3(6x,8hrotation), /1x)
2004     format(1h0,i6,i8,6e14.5 /(7x,i8,6e14.5))
c
         end


         subroutine xload(neq,11,b)
c          implicit real*8(a-h,o-z)
         dimension b(neq,11)
         rewind 3
         read(3) b
c         write(6,*)' xload neq 11',neq,11
         do 1 i=1,neq
c         bb(i)=b(i,11)
c         write(6,*) b(i,1),'bb(i) xload'
1        continue
         return
         end
c
c*************************************************************************
      Forcesub ROW9(A,B,MAXA,IROWL,ICOLH,NEQ,NEQP1,NTERMS,IFLAG
     +  ,jops,lc)    of NNP ident ME

        REAL A(NTERMS),B(NEQ,lc)
        INTEGER MAXA(NEQP1),IROWL(NEQ),ICOLH(NEQ)
        INTEGER jops
        Private INTEGER I,J,K,L,IM1,IC1,IBOT,ICOL,ICOLP,ITOP,JROW,KM1
        Private INTEGER JM1,JM2,JM3,JM4,JM5,JM6,JM7,JM8,jm9,IDIV,IDIV1
        Private INTEGER JTOP,JBOT,ICOPY,jj1 ,jjrow
        Private REAL XMULT1,XMULT2,XMULT3,XMULT4,XMULT5,XMULT6,XMULT7,
     +               XMULT8,TEMP,XINV,SUM
```

```
      Async REAL   X(10001)
      End Declarations
c     write(6,*)' row9 starts ++++++'

c       Barrier
c          if(me.eq.3) then
c       do 198 i1=1,nterms
c198    write(6,*)' a vector at the beginning of row9',a(i1)
c            write(6,*)'b,maxa,irwl,icolh'
c        do 199 i1=1,neq
c199      write(6,*)b(i1,1),maxa(i1),irowl(i1),icolh(i1)
c              End barrier
c               endif
C     ..............................................................

      IF(IFLAG.EQ.1) THEN

        Presched  DO 9 I = 1, NEQ
         Void X(I)
9       End Presched Do
c      write(*,*) 'void has been completed'
         jops = 0
      Barrier
        jops = 0
         A(1) = SQRT(A(1))
         XINV = 1.0/A(1)
CDIR$ IVDEP
         DO 20 K = 1, IROWL(1)
            A(K+1) = XINV * A(K+1)
 20      CONTINUE
c        write(*,*) 'first row has been processed'
         jops = jops + irowl(1)+2
         Produce X(1)=a(1)
c        write(*,*) 'first void has been unvoided'
       End Barrier


C.....DECOMPOSED STIFFNESS MATRIX PHASE

       Presched DO 100 I = 2, NEQ

c         TAKES CARE OF ROWS ONE BY ONE
          iml = maxa(i)
          icl = icolh(i)
c      indices calculation for using the modification factor
c      from the upper segment of column-height.
          ibot = i - 9*( (i-1)/9 )
          icol = icl - ibot + 1
          icolp= icol/9
          itop = icol - 9*icolp
c ....indices calculation for modifcation by itop elements.
          jrow = i - icl
          jml  = maxa(jrow) + icl
          jjrow=irowl(jrow)
c         write(*,*) 'iml,icl,ibot,icol,icolp,itop,jrow,jml'
```

```
c        write(*,*) iml,icl,ibot,icol,icolp,itop,jrow,jml

         IF (ITOP. GE. 1 ) THEN

             ICOPY = JROW + ITOP -1
c            If (Isfull(x(icopy))) go to 331
             Copy X(ICOPY) INTO TEMP
c       write(*,*) 'the statement icop=',icop,'has been checked'
         ENDIF

331       go to (101,102,103,104,105,106,107,108), itop

C...........................................................
          go to 150

CDIR$ IVDEP
 101       do 111 k = 1, jjrow-icl+1
             kml = k -1
             a(iml+kml)  = a(iml+kml)  -a(jml)*a(jml+kml)
  111      continue
          go to 150

 102       jm2 = jml + jjrow
CDIR$ IVDEP
           do 112 k = 1, jjrow-icl+1
             kml = k -1
             a(iml+kml)  = a(iml+kml)  -a(jml)*a(jml+kml)
     +                     - a(jm2)*a(jm2+kml)
  112      continue
          go to 150

 103       jm2 = jml + jjrow
           jm3 = jm2 + jjrow -1
CDIR$ IVDEP
           do 113 k = 1, jjrow -icl+1
             kml = k -1
             a(iml+kml)  = a(iml+kml)  - a(jml)*a(jml+kml)
     +                    -a(jm2)*a(jm2+kml)  -a(jm3)*a(jm3+kml)
  113      continue
          go to 150

 104       jm2 = jml + jjrow
           jm3 = jm2 + jjrow -1
           jm4 = jm3 + jjrow -2
CDIR$ IVDEP
           do 114 k = 1, jjrow -icl+1
             kml = k -1
             a(iml+kml)  = a(iml+kml)  - a(jml)*a(jml+kml)
     +                    -a(jm2)*a(jm2+kml)  -a(jm3)*a(jm3+kml)
     +                    -a(jm4)*a(jm4+kml)
  114      continue
          go to 150

 105       jm2 = jml + jjrow
           jm3 = jm2 + jjrow -1
```

```
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
CDIR$ IVDEP
            do 115 k = 1, jjrow -icl+1
              km1 = k -1
              a(im1+km1) = a(im1+km1) - a(jm1)*a(jm1+km1)
     +                   -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                   -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
  115       continue
           go to 150

 106        jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
            jm6 = jm5 + jjrow -4
CDIR$ IVDEP
            do 116 k = 1, jjrow -icl+1
              km1= k -1
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +                   -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                   -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
     +                   -a(jm6)*a(jm6+km1)
  116       continue
           go to 150

 107        jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
            jm6 = jm5 + jjrow -4
            jm7 = jm6 + jjrow -5
CDIR$ IVDEP
            do 117 k = 1, jjrow -icl+1
              km1 = k -1
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +                   -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                   -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
     +                   -a(jm6)*a(jm6+km1) -a(jm7)*a(jm7+km1)
  117       continue
           go to 150
 108        jm2 = jm1 + jjrow
            jm3 = jm2 + jjrow -1
            jm4 = jm3 + jjrow -2
            jm5 = jm4 + jjrow -3
            jm6 = jm5 + jjrow -4
            jm7 = jm6 + jjrow -5
            jm8 = jm7 + jjrow -6
CDIR$ IVDEP
            do 118 k = 1, jjrow -icl+1
              km1 = k -1
              a(im1+km1) = a(im1+km1) -a(jm1)*a(jm1+km1)
     +                   -a(jm2)*a(jm2+km1) -a(jm3)*a(jm3+km1)
     +                   -a(jm4)*a(jm4+km1) -a(jm5)*a(jm5+km1)
     +                   -a(jm6)*a(jm6+km1) -a(jm7)*a(jm7+km1)
```

```
     +                      -a(jm8)*a(jm8+km1)
  118      continue
           go to 150

C..............................................................

 150       jops = jops + itop*(jjrow -icl+2)*2
           ll =   3
           idiv =  1
           if (icolp.le.ll) then
              ll =icolp
              idivl=1
           else
              idivl=icolp-ll+1
           endif
           jtop = icl
           jbot = icl-itop+1

c          write(*,*) 'll,idiv,idivl,jtop,jbot'
c          write(*,*) ll,idiv,idivl,jtop,jbot
           do 10 l = 1,  ll
              jtop = jtop - itop
              jbot = jbot - 9*idivl
              itop = 9*idivl
              idivl = idiv

               if (l.eq.ll) then
              icopy = i - 1
              else
              icopy = i -jbot +ibot-1
                endif
c      write(*,*) 'jtop,jbot,itop,idivl',jtop,jbot,itop,idivl,icop
c           If (lsfull(x(icopy))) go to 332
           Copy X(icopy) into temp

c          write(*,*) 'icop has been cleared'
332           do 200 j = jtop, jbot, -9
                    JJ1 = I-J
                    jjrow = irowl(jj1)
                    jml = maxa(jj1) + j
                    jm2 = jml + jjrow
                    jm3 = jm2 + jjrow -1
                    jm4 = jm3 + jjrow -2
                    jm5 = jm4 + jjrow -3
                    jm6 = jm5 + jjrow -4
                    jm7 = jm6 + jjrow -5
                    jm8 = jm7 + jjrow -6
                    jm9 = jm8 + jjrow -7
c                   xmultl = a(jml)
c                   XMULT2 = A(JM2)
c                   XMULT3 = A(JM3)
c                   XMULT4 = A(JM4)
c                   xmult5 = a(jm5)
c                   xmult6 = a(jm6)
c                   xmult7 = a(jm7)
```

```
c                      xmult8 = a(jm8)
c                      xmult9 = a(jm9)
CDIR$ IVDEP
              DO 300 K = 1, jjrow -J +1
                     KM1 = K -1
                     A(im1+km1)  = A(im1+km1)
     .                            -a(jm1)*a(jm1+km1)   -a(jm2)*a(jm2+km1)
     .                            -a(jm3)*a(jm3+km1)   -a(jm4)*a(jm4+km1)
     .                            -a(jm5)*a(jm5+km1)   -a(jm6)*a(jm6+km1)
     .                            -a(jm7)*a(jm7+km1)   -a(jm8)*a(jm8+km1)
     .                            -a(jm9)*a(jm9+km1)
 300            CONTINUE
              jops = jops + 18*( jjrow -j+1)

 200          CONTINUE
 10         continue

              l1=i-1
c              If (Isfull(x(l1))) go to 333
              Copy x(l1) into temp

c             write(*,*) 'l1 has been cleared',l1
333           go to (201,202,203,204,205,206,207,208) ibot-1
                 go to 250
c.............................................................

 201               jjrow = irowl(i-1)
                   jm1 = maxa(i-1) +1
CDIR$ IVDEP
                   DO 211 K = 1, jjrow
                      KM1 = K -1
                      A(IM1+KM1)  = A(IM1+KM1)  - a(jm1)* A(JM1 +KM1)
  211              CONTINUE
              go to 250

  202                 jjrow = irowl(i-2)
                      jm1 = maxa(i-2) +2
                      JM2 = jm1 + jjrow
CDIR$ IVDEP
                      DO 212 K = 1, jjrow -1
                         KM1 = K -1
                         A(IM1+KM1)  = A(IM1+KM1)  - a(jm1)*a(jm1+km1)
     .                                 -A(jm2)*A(JM2+KM1)
  212                 CONTINUE
              go to 250

 203                  jjrow = irowl(i-3)
                      jm1 = maxa(i-3) + 3
                      JM2 = jm1 + jjrow
                      JM3 = jm2 + jjrow -1
CDIR$ IVDEP
                      DO 213 K = 1, jjrow -2
                         KM1=K -1
                         A(IM1+KM1)  = A(IM1+KM1)  -A(jm1)*A(JM1+KM1)
     .                                 -a(jm2)*A(JM2+KM1)-a(jm3)*A(JM3+KM1)
```

```
  213           CONTINUE
          go to 250

 204            jjrow = irowl(i-4)
                jml = maxa(i-4) + 4
                jm2 = jml + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
CDIR$ IVDEP
                do 214 k = 1,jjrow  -3
                   kml = k -1
                   a(iml+kml) = a(iml+kml) -a(jml)*a(jml+kml)
     .                      -a(jm2)*a(jm2+kml)-a(jm3)*a(jm3+kml)
     .                      -a(jm4)*a(jm4+kml)
  214           continue
          go to 250

  205           jjrow = irowl(i-5)
                jml = maxa(i-5) + 5
                jm2 = jml + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
                jm5 = jm4 + jjrow -3
CDIR$ IVDEP
                do 215 k = 1, jjrow -4
                   kml = k -1
                   a(iml+kml) = a(iml+kml) -a(jml)*a(jml+kml)
     .                      -a(jm2)*a(jm2+kml)-a(jm3)*a(jm3+kml)
     .                      -a(jm4)*a(jm4+kml)-a(jm5)*A(jm5+kml)
  215           continue
          go to 250

 206            jjrow = irowl(i-6)
                jml = maxa(i-6) +6
                jm2 = jml + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
                jm5 = jm4 + jjrow -3
                jm6 = jm5 + jjrow -4
CDIR$ IVDEP
                do 216 k = 1, jjrow -5
                   kml = k -1
                   a(iml+kml) = a(iml+kml) -a(jml)*a(jml+kml)
     .                      -a(jm2)*a(jm2+kml)-a(jm3)*a(jm3+kml)
     .                      -a(jm4)*a(jm4+kml)-a(jm5)*a(jm5+kml)
     .                      -a(jm6)*a(jm6+kml)
  216           continue
          go to 250

 207            jjrow = irowl(i-7)
                jml = maxa(i-7)+7
                jm2 = jml + jjrow
                jm3 = jm2 + jjrow -1
                jm4 = jm3 + jjrow -2
                jm5 = jm4 + jjrow -3
```

```
                 jm6 = jm5 + jjrow -4
                 jm7 = jm6 + jjrow -5
CDIR$ IVDEP
                 do 217 k = 1, jjrow -6
                    km1 = k -1
                    a(im1+km1)  = a(im1+km1)  -a(jm1)*a(jm1+km1)
                                 -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .                           -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     .                           -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
     .
  217            continue
               go to 250

 208             jjrow =irowl(i-8)
                 jm1 = maxa(i-8) + 8
                 jm2 = jm1 + jjrow
                 jm3 = jm2 + jjrow -1
                 jm4 = jm3 + jjrow -2
                 jm5 = jm4 + jjrow -3
                 jm6 = jm5 + jjrow -4
                 jm7 = jm6 + jjrow -5
                 jm8 = jm7 + jjrow -6
CDIR$ IVDEP
                 do 218 k = 1, jjrow -7
                    km1 = k -1
                    a(im1+km1)  = a(im1+km1)  - a(jm1)*a(jm1+km1)
                                 -a(jm2)*a(jm2+km1)-a(jm3)*a(jm3+km1)
     .                           -a(jm4)*a(jm4+km1)-a(jm5)*a(jm5+km1)
     .                           -a(jm6)*a(jm6+km1)-a(jm7)*a(jm7+km1)
     .                           -a(jm8)*a(jm8+km1)
     .
  218            continue
               go to 250
C.........................................................

250      jops = jops + 2*(ibot-1)*(jjrow -ibot +2)
         A(IM1) =SQRT(A(IM1))
c        WRITE(6,*) 'A(',IM1,')=',A(IM1)
         XINV = 1.0/A(IM1)
CDIR$ IVDEP
         DO 260  K = 1, IROWL(I)
            A(IM1+K) = XINV *A(IM1+K)
  260    CONTINUE
         jops = jops + irowl(i) +2
         Produce X(I) = A(IM1)
c        write(*,*) 'row',i,'is cleared'
c        WRITE(6,*)  (A(IM1+L),L=1,IROWL(I))
 100     End Presched Do
         ELSE
C.....FORWARD REDUCTION

         do 196 lo=1,lc
        Barrier
           jops = 0
        DO 510 I = 1,NEQ
           B(I,lo) = B(I,lo)/A(MAXA(I))
           SUM = B(I,lo)
```

```
            IM1 =MAXA(I)
CDIR$ IVDEP
            DO 520 J = I+1, I+IROWL(I)
               B(J,lo) = B(J,lo) - SUM* A(IM1+J-I)
520         CONTINUE
            jops = jops + 2*(irowl(i))+ 2
510      CONTINUE
C........BACK SUBSTITUTION
        B(NEQ,lo) = B(NEQ,lo)/A(MAXA(NEQ))
        jops = jops +1
        DO 1010 I = NEQ-1,1,-1
           SUM = 0.0
CDIR$ IVDEP
           DO 1020 J =I+1, IROWL(I)+I
             SUM=SUM+ A(MAXA(I)+J-I)*B(J,lo)
 1020      CONTINUE
           B(I,lo) =(B(I,lo)-SUM)/A(MAXA(I))
    jops = jops + 2*(irowl(i)) +2
 1010   CONTINUE
      End Barrier
196     continue
      ENDIF
c      do 129 ii=1,neq
c129    write(6,*)b(ii,1)
       Barrier
        rewind 2
        write(2) ((b(i,lo),i=1,neq),lo=1,lc)
c        write(6,*)' sub row9 ends....++++++'
       End barrier
      RETURN
      END
      subroutine column
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
cnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
c      specify the level of loop unrolling
c      modify the row length for loop unrol purpose
       nnblock=neq/loopur
c       write(6,*)' nnblock..neq,loopur',nnblock,neq,loopur
       leftov=neq-(nnblock*loopur)
c
       maxcol=0
       do 5 i=1,nnblock
       istart=(i-1)*loopur+1
       iend=i*loopur
       do 6 jrow=istart,iend
       jcol=jrow+irowl(jrow)-1
       if(jcol.gt.maxcol)maxcol=jcol
6      continue
c      now increase each row length for loopur purpose
       do 7 jrow=istart,iend
7      irowl(jrow)=maxcol-jrow+1
5      continue
c      now take care of the left over row
       istart=nnblock*loopur+1
```

```
      iend=neq
      do 8 jrow=istart,iend
8     irowl(jrow)=neq-jrow+1
c88888888888888888888888888888888888888888 column height
c     to find the column height
cn      icolh(neq)=0
cn      do 91 i=neq-1,1,-1
cn      irl=irowl(i)
cn      do 92 j=i,i+irl-1
cn92    icolh(j)=j-i
cn91    continue
c*****************************************
c     find the location of the diagonal terms
      naxa(1)=1
      do 11 i=2,neq
11    naxa(i)=naxa(i-1)+irowl(i-1)
c      find the total number of terms
       nterms=naxa(neq)
cnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnnn
c     to find the column height
      icolh(neq)=0
      do 91 i=neq-1,1,-1
      irl=irowl(i)
      do 92 j=i,i+irl-1
92    icolh(j)=j-i
91    continue
c     update row length not to include the diagonals
c      do 17 i=1,neq
c17    irowl(i)=irowl(i)-1
c
         return
         end
c**************** beam subroutines
      SUBROUTINE BEAM
C
C
C     CALLS?  TEAM,STRSC
C     CALLED BY?  ELTYPE
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /JUNK/ LT,LH,L,IPAD,SIG(20),N6,N7,N8,N9,N10,IFILL(381)
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
C***     COMMON A(1)
      COMMON /one/A(1)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
c      COMMON A(7100)
C
      IF(NPAR(1).EQ.0) GO TO 500
      N5A=N5+NUMNP
      N6=N5+NPAR(5) + NUMNP
      N7=N6+NPAR(5)
      N8=N7+NPAR(5)
      N9=N8+12*NPAR(4)
      N10=N9+6*NPAR(3)
```

```
      N11=N10+NPAR(5)
      IF(N11.GT.MTOT) CALL ERROR(N11-MTOT)
C
      CALL TEAM(NPAR(2),NPAR(3),NPAR(4),NPAR(5),A(N1),A(N2),A(N3),
     1          A(N4),A(N5A),A(N6),A(N7),A(N8),A(N9),A(N10),
     2          NUMNP,MBAND)
C
      RETURN
C
  500 WRITE (6,2002)
      NUME=NPAR(2)
      numee=nume
       neqq=neq
      DO 800 MM=1,NUME
      CALL STRSC (A(N1),A(N3),NEQ,0)
      WRITE (6,2001)
      DO 800 L=LT,LH
      CALL STRSC (A(N1),A(N3),NEQ,1)
      WRITE(6,3002) MM,L,(SIG(I),I=1,12)
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,(SIG(I),I=1,12)
  800 CONTINUE
      RETURN
 2001 FORMAT (/)
 2002 FORMAT(/29H1.....BEAM FORCES AND MOMENTS//
     . 10H0BEAM LOAD 5X 5HAXIAL  2(7X,5HSHEAR),5X 7HTORSION
     .  2(5X,7HBENDING)/ 10H  NO.  NO.  8X  2HR1  10X  2HR2  10X
     .  2HR3 10X 2HM1 10X 2HM2 10X 2HM3)
 3002 FORMAT (I5,I4,1PE11.3,5E12.3/8X,6E12.3/)
      END
      SUBROUTINE ERROR(N)
      WRITE (6,2000) N
 2000 FORMAT (// 20H STORAGE EXCEEDED BY  I6)
      STOP
      END
      SUBROUTINE NEWBM(E,G,RO,WGHT,COPROP,SFT,NUMFIX,NUMETP)
C
C     CALLED BY?  TEAM
C
C     FORM NEW BEAM STIFFNESS
C
      DIMENSION E(1),G(1),RO(1),COPROP(NUMETP,1),SFT(NUMFIX,1),WGHT(1)
      COMMON/EM/LM(24),ND,NS,ASA(24,24),RF(24,4),XM(24),SA(12,24),
     1 SF(12,4),XWT(24),IFILL(3000)
      COMMON /NEWB/ LC(4),T(3,3),JK(6),MELTYP,MATTYP,DL
      DIMENSION R(12),S(12,12),C(12)
C
      DO 5 I=1,12
      DO 5 J=1,12
    5 S(I,J)=0.0D0
      AX=COPROP(MELTYP,1)
       AY=COPROP(MELTYP,2)
       AZ=COPROP(MELTYP,3)
      AAX=COPROP(MELTYP,4)
```

```
      AAY=COPROP(MELTYP,5)
      AAZ=COPROP(MELTYP,6)
      SHFY=0.0
      SHFZ=0.0
      ZY=E(MATTYP)/(DL*DL)
      EIY=ZY*AAY
      EIZ=ZY*AAZ
      IF(AY.NE.0.0) SHFY=6.*EIZ/(G(MATTYP)*AY)
      IF(AZ.NE.0.0) SHFZ=6.*EIY/(G(MATTYP)*AZ)
      COMMY=EIY/(1.+2.*SHFZ)
      COMMZ=EIZ/(1.+2.*SHFY)
C
C        FIXED END FORCES IN LOCAL COORDS
C
      DO 73 N=1,4
      M=LC(N)
      IF (M.GT.0)  GO TO 71
      DO 70 I=1,12
   70 SF(I,N)=0.
      GO TO 73
   71 DO 72 I=1,12
   72 SF(I,N)=SFT(M,I)
   73 CONTINUE
C
C     FORM ELEMENT STIFFNESS IN LOCAL COORDINATES
C
      S(1,1)= E(MATTYP)* AX/DL
      S(4,4)= G(MATTYP)*AAX/DL
      S(2,2)= COMMZ*12./DL
      S(3,3)= COMMY*12./DL
      S(5,5)= COMMY* 4.*DL*(1.+0.5*SHFZ)
      S(6,6)= COMMZ* 4.*DL*(1.+0.5*SHFY)
      S(2,6)= COMMZ* 6.
      S(3,5)=-COMMY* 6.
      DO 102 I=1,6
      J=I+6
  102 S(J,J)=S(I,I)
      DO 104 I=1,4
      J=I+6
  104 S(I,J)=-S(I,I)
      S(6,12)= S(6,6)*(1.-SHFY)/(2.+SHFY)
      S(5,11)= S(5,5)*(1.-SHFZ)/(2.+SHFZ)
      S(2,12)= S(2,6)
      S(6, 8)=-S(2,6)
      S(8,12)=-S(2,6)
      S(3,11)= S(3,5)
      S(5, 9)=-S(3,5)
      S(9,11)=-S(3,5)
      DO 106 I=2,12
      K=I-1
      DO 106 J=1,K
  106 S(I,J)=S(J,I)
C
C     MODIFY ELEMENT STIFFNESS AND ELEMENT FIXED END FORCES FOR KNOWN
C     ZERO MEMBER END FORCES.
```

```
C
      IF ((JK(1)+JK(2)).EQ.0) GO TO 145
      DO 140 K=1,2
      KK=JK(K)
      KD=100000
      I1=6*(K-1)+1
      I2=I1+5
      DO 140 I=I1,I2
      IF (KK.LT.KD)  GO TO 140
      SII=S(I,I)
      DO 125 N=1,12
  125 R(N)=S(I,N)
      DO 130 M=1,12
      C(M)=S(M,I)/SII
      DO 130 N=1,12
  130 S(M,N)=S(M,N)-C(M)*R(N)
      DO 135 N=1,4
      SFI=SF(I,N)
      DO 135 M=1,12
  135 SF(M,N)=SF(M,N)-C(M)*SFI
  136 KK=KK-KD
  140 KD=KD/10
  145 CONTINUE
C
C     OBTAIN SA(12,12) RELATING ELEMENT END FORCES (LOCAL) AND
C     JOINT DISPLACEMENTS (GLOBAL).
C
      DO 31 I=1,12
      DO 31 J=1,24
   31 SA(I,J)=0.0D0
      DO 150 LA=1,10,3
      LB=LA+2
      DO 150 MA=1,10,3
      MB=MA-1
      DO 150 I=LA,LB
      DO 150 JM=1,3
      J=JM+MB
      XX=0.
      DO 151 K=1,3
  151 XX=XX+S(I,K+MB)*T(K,JM)
  150 SA(I,J)=XX
C
C     ELEM STIFF ASA(12,12) AND FIXED END FORCES RF(12) IN GLOBAL COORDS
C
      DO 32 I=1,24
      DO 32 J=1,24
   32 ASA(I,J)=0.0D0
      DO 160 LA=1,10,3
      LB=LA-1
      DO 160 MA=1,10,3
      MB=MA+2
      DO 160 IL=1,3
      I=IL+LB
      DO 160 J=MA,MB
      XX=0.
```

```
      DO 161 K=1,3
  161 XX=XX+T(K,IL)*SA(K+LB,J)
  160 ASA(I,J)=XX
C
      DO 165 LA=1,10,3
      LB=LA-1
      DO 165 IL=1,3
      I=IL+LB
      DO 165 N=1,4
      XX=0.
      DO 162 K=1,3
  162 XX=XX-T(K,IL)*SF(K+LB,N)
  165 RF(I,N)=XX
C
C     FORM MASS AND GRAVITY LOAD MATRIX
C
      XXM=RO(MATTYP)*AX*DL/2.
      WTM=WGHT(MATTYP)*AX*DL/2.
      DO 180 M=1,3
      XWT(M)=WTM
      XWT(M+3)=0.
      XWT(M+9)=0.
      XWT(M+6)=WTM
      XM(M)=XXM
      XM(M+3)=0.
      XM(M+9)=0.
  180 XM(M+6)=XXM
      RETURN
      END
      SUBROUTINE SLAVE (X,Y,Z,ID,NUMNP,NI,NJ)
C
C     CALLED BY?  TEAM
C
C     PERFORMS SLAVE...MASTER DISPLACEMENT TRANSFORMATION
C       ( FOR NODES CONNECTED TO BEAM ELEMENTS ONLY)
C
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,1)
      COMMON /EM/ LM(24),ND,NS,S(24,24),R(96),XM(24),SA(12,24)  ,TT(12,4)
     1          ,IFILL(3048)
      COMMON /EXTRA/ MODEX,NT8
C     DETERMINE REQUIRED TRANSLATION DEGREES OF FREEDOM
C
      DO 54 NF=1,12,6
      NOD=NI
      IF (NF.EQ.7) NOD=NJ
      DO 30 K=1,3
      I=K+NF-1
      IF (LM(I).GE.0) GO TO 30
      M=-LM(I)
      LM(I)=ID(M,K)
      IF(K-2) 35,45,55
   35 D1=-(Y(NOD)-Y(M))
      D2=    Z(NOD)-Z(M)
      LM(ND+1)=ID(M,6)
      LM(ND+2)=ID(M,5)
```

```
      GO TO 50
   45 D1=-(Z(NOD)-Z(M))
      D2=   X(NOD)-X(M)
      LM(ND+1)=ID(M,4)
      LM(ND+2)=ID(M,6)
      GO TO 50
   55 D1=-(X(NOD)-X(M))
      D2=   Y(NOD)-Y(M)
      LM(ND+1)=ID(M,5)
      LM(ND+2)=ID(M,4)
   50 CONTINUE
      IF  (MODEX.EQ.1) GO TO 80
C     TRANSFORMATION...ARRAYS INCREASE IN SIZE
C
      DO 60 II=1,ND
      S(ND+1,II)=S(I,II)*D1
      S(ND+2,II)=S(I,II)*D2
      S(II,ND+1)  =  S(II,I)  *D1
      S(II,ND+2)  =  S(II,I)  *D2
   60 CONTINUE
      XM(ND + 1)  =  XM(I)*D1*D1
      XM(ND + 2)  =  XM(I)*D2*D2
C
      DO 70 II=1,NS
      SA(II,ND+1)=SA(II,I)*D1
   70 SA(II,ND+2)=SA(II,I)*D2
C
      S(ND+1,ND+1)=S(I,I)*D1**2
      S(ND+2,ND+2)=S(I,I)*D2**2
      S(ND+1,ND+2)=S(I,I)*D1*D2
      S(ND+2,ND+1)=S(ND+1,ND+2)
   80 ND = ND + 2
   30 CONTINUE
C
C     SET ROTATIONS
C
      DO 54 J=1,3
      K=NF+J+2
      IF(LM(K).GE.0)  GO TO 54
      M=-LM(K)
      LM(K)=ID(M,J+3)
   54 CONTINUE
C
      RETURN
      END
      SUBROUTINE TEAM(NBEAM,NUMETP,NUMFIX,NUMMAT,ID,X,Y,Z,E,G,RO,
     1 SFT,COPROP,WGHT,NUMNP,MBAND)
C
C     CALLS?  NEWBM,SLAVE,CALBAN
C     CALLED BY?  BEAM
C
C     FORMS 3-D BEAM STIFFNESS AND STRESS ARRAYS
C
      COMMON/EM/LM(24),ND,NS,ASA(24,24),RF(24,4),XM(24),SA(12,24),
     1 SF(12,4),XWT(24),IFILL(3000)
```

```
      COMMON /NEWB/ LC(4),T(3,3),JK(6),MELTYP,MATTYP,DL
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,1),E(1),G(1),SFT(NUMFIX,1)
     1 ,COPROP(NUMETP,1),RO(1),EMUL(3,4),WGHT(1)
      DIMENSION ILC(4),TI(3,3),TJ(3,3),STIF(722),TS(2,2),LS(4)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      EQUIVALENCE (STIF(1),LM(1))
C
C
C     INITIALIZATION
C
      WRITE (6,2005) NBEAM,NUMETP,NUMFIX,NUMMAT
      N=0
      DO 5 I=1,1058
    5 STIF(I)=0.
C
C     READ AND PRINT MATERIAL PROPERTY DATA
C
      WRITE (6,2001)
      DO 10 I=1,NUMMAT
      READ (5,1001) N,E(N),G(N),RO(N),WGHT(N)
      WRITE(6,2002) N,E(N),G(N),RO(N),WGHT(N)
   10 G(N)=0.5*E(N)/(1.+G(N))
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) (E(N),G(N),RO(N),N=1,NUMMAT)
C
C     READ AND PRINT GEOMETRIC PROPERTIES OF COMMON ELEMENTS.
C
      WRITE (6,2003)
      DO 30 I=1,NUMETP
      READ  (5,1002) N,(COPROP(N,J),J=1,6)
      IF((COPROP(N,1).NE.0.0).AND.(COPROP(N,4).NE.0.0).AND.
     1   (COPROP(N,5).NE.0.0).AND.(COPROP(N,6).NE.0.0))  GO TO 20
      WRITE (6,2013)
      STOP
   20 WRITE (6,2004) N,(COPROP(N,J),J=1,6)
   30 CONTINUE
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) ((COPROP(N,J),J=1,6),N=1,NUMETP)
C
C     ELEMENT LOAD MULTIPLIERS
C
      READ  (5,1006) ((EMUL(I,J),J=1,4),I=1,3)
      WRITE (6,2006) ((EMUL(I,J),J=1,4),I=1,3)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) ((EMUL(I,J),J=1,4),I=1,3)
C
C     READ AND PRINT FIXED END FORCES IN LOCAL COORDINATES
C
      IF(NUMFIX .EQ. 0) GO TO 56
      WRITE (6,2010)
```

```
      DO 55 I=1,NUMFIX
      READ  (5,1005) N,(SFT(N,J),J=1,12)
   55 WRITE (6,2011) N,(SFT(N,J),J=1,12)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) ((SFT(N,J),J=1,12),N=1,NUMFIX)
   56 CONTINUE
C
C     READ AND PRINT ELEMENT DATA. GENERATE MISSING INPUT.
C
      WRITE (6,4000)
      L=0
   60 KKK=0
      READ (5,3000) INEL,INI,INJ,INK,IMAT,IMEL,ILC,INELKI,INELKJ,INC
      IF (INEL.NE.1) GO TO 15
      NI=INI
      NJ=INJ
      NK=INK
   15 IF (INC.EQ.0) INC=1
   65 L=L+1
      KKK=KKK+1
      ML=INEL-L
      IF (ML) 66,67,68
   66 WRITE (6,4003) INEL
      STOP
   67 NEL=INEL
       NI    =INI
       NJ    =INJ
      NK=INK
       MATTYP=IMAT
       MELTYP=IMEL
      DO 90 I=1,4
   90 LC(I)=ILC(I)
       NLOAD=LC(1)+LC(2)+LC(3)+LC(4)
       NEKODI=INELKI
       NEKODJ=INELKJ
      DO 91 I=1,3
   91 T(2,I)=TI(2,I)
      GO TO 69
   68 NEL=INEL-ML
       NI    =IN+KKK*INCR
       NJ    =JN+KKK*INCR
   69 CONTINUE
      WRITE (6,4001) NEL,NI,NJ,NK,MATTYP,MELTYP,LC,NEKODI,NEKODJ
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) NEL,NI,NJ,NK,MATTYP,MELTYP,LC,NEKODI,NEKODJ
C
   74 DX=X(NJ)-X(NI)
      DY=Y(NJ)-Y(NI)
      DZ=Z(NJ)-Z(NI)
      DL=SQRT(DX*DX+DY*DY+DZ*DZ)
      IF(DL) 75,75,76
   75 WRITE (6,4005) NEL
      STOP
```

```
C
C     FORM GLOBAL TO LOCAL COORDINATE TRANSFORMATION.
C
   76 T(1,1)=DX/DL
      T(1,2)=DY/DL
      T(1,3)=DZ/DL
C
C     COMPUTE DIRECTION COSINES OF LOCAL Y-AXIS
C
      A1=X(NJ)-X(NI)
      A2=Y(NJ)-Y(NI)
      A3=Z(NJ)-Z(NI)
      B1=X(NK)-X(NI)
      B2=Y(NK)-Y(NI)
      B3=Z(NK)-Z(NI)
      AA=A1*A1+A2*A2+A3*A3
      AB=A1*B1+A2*B2+A3*B3
      U1=AA*B1-AB*A1
      U2=AA*B2-AB*A2
      U3=AA*B3-AB*A3
      UU=U1*U1+U2*U2+U3*U3
      UU=SQRT(UU)
      IF (UU.GT.0.) GO TO 40
      WRITE (6,4002) INEL
      STOP
   40 CONTINUE
      IF(MODEX.EQ.1) GO TO 185
      T(2,1)=U1/UU
      T(2,2)=U2/UU
      T(2,3)=U3/UU
      T(3,1)=T(1,2)*T(2,3)-T(1,3)*T(2,2)
      T(3,2)=T(1,3)*T(2,1)-T(1,1)*T(2,3)
      T(3,3)=T(1,1)*T(2,2)-T(1,2)*T(2,1)
C
C     CHECK IF NEW STIFFNESS NEEDED
C
      IF (NEL.GE.1) GO TO 80
      IF (ABS(DS-DL) .GT. DL/100.) GO TO 80
      IF ((MT.NE.MATTYP).OR.(ME.NE.MELTYP)) GO TO 80
      IF ((JK(1).NE.NEKODI).OR.(JK(2).NE.NEKODJ)) GO TO 80
      DO 81 I=1,4
      IF (LS(I).NE.LC(I)) GO TO 80
   81 CONTINUE
      DO 82 I=1,2
      DO 82 J=1,2
      IF (ABS(TS(I,J)-T(I,J)) .GT. ABS(T(I,J)/100.)) GO TO 80
   82 CONTINUE
      GO TO 185
C
   80 DS=DL
      MT=MATTYP
      ME=MELTYP
      DO 77 I=1,2
      DO 77 J=1,2
   77 TS(I,J)=T(I,J)
```

```
      DO 78 I=1,4
   78 LS(I)=LC(I)
      JK(1)=NEKODI
      JK(2)=NEKODJ
C
C     FORM NEW STIFFNESS
C
      CALL NEWBM(E,G,RO,WGHT,COPROP,SFT,NUMFIX,NUMETP)
C
C     ADD GRAVITY LOADING ... POINT LOADS ONLY COMPUTED
C
      DO 180 I=1,3
      DO 180 J=1,4
      RF(I,J)=RF(I,J)+EMUL(I,J)*XWT(I)
  180 RF(I+6,J)=RF(I+6,J)+EMUL(I,J)*XWT(I+6)
C
C     FORM ELEMENT LOCATION MATRIX
C
  185 CONTINUE
      DO 170 M=1,6
      LM(M)=ID(NI,M)
      LM(M+12)=0
      LM(M+18)=0
  170 LM(M+6)=ID(NJ,M)
C
      NS=12
      ND=12
C
C     TRANSFORM TO MASTER DEGREES OF FREEDOM
C
C
      CALL SLAVE(X,Y,Z,ID,NUMNP,NI,NJ)
C
C     WRITE ELEMENT INFORMATION ON TAPE
C
      NDM=24
      CALL CALBAN (MBAND,NDIF,LM,XM,ASA,RF,ND,NDM,NS)
      IF(MODEX.EQ.1) GO TO 300
      WRITE (1) ND,NS,(LM(I),I=1,ND),((SA(I,J),I=1,NS),J=1,ND),
     1 ((SF(I,J),I=1,NS),J=1,4)
C
C         CHECK FOR LAST ELEMENT
C
  300 IF(NBEAM-NEL) 66,500,260
  260 CONTINUE
      IF (ML.GT.0)  GO TO 65
       IN     =INI
       JN     =INJ
      INCR=INC
      GO TO 60
  500 RETURN
C
 1001 FORMAT(I5,4F10.0)
 1002 FORMAT(I5,6F10.0)
 1005 FORMAT(I5,6F10.0/F15.0,5F10.0)
```

```
 1006 FORMAT (4F10.0)
C
 2001 FORMAT (/// 20H MATERIAL PROPERTIES, // 5X,8HMATERIAL,8X,
     1        7HYOUNG*S,6X,9HPOISSON*S,11X,4HMASS,9X,6HWEIGHT, / 7X,
     2        6HNUMBER,8X,7HMODULUS,10X,5HRATIO,2(8X,7HDENSITY), / 1X)
C
 2002 FORMAT (8X,I5,E15.4,F15.4,2E15.4)
 2003 FORMAT (/// 26H BEAM GEOMETRIC PROPERTIES, // 5X,7HSECTION,3X,
     1        10HAXIAL AREA,2(3X,10HSHEAR AREA),6X,7HTORSION,2(6X,
     2        7HINERTIA),/ 6X,6HNUMBER,9X,4HA(1),9X,4HA(2),9X,4HA(3),
     3        9X,4HJ(1),9X,4HI(2),9X,4HI(3), / 1X)
 2004 FORMAT (7X,I5,6E13.4)
 2005 FORMAT (34H13 / D   B E A M   E L E M E N T S, ///
     .  36H  NUMBER OF BEAMS                  =,I5/
     .  36H  NUMBER OF GEOMETRIC PROPERTY SETS=,I5/
     .  36H  NUMBER OF FIXED END FORCE SETS   =,I5/
     .  36H  NUMBER OF MATERIALS              =,I5)
 2006 FORMAT(///25H ELEMENT LOAD MULTIPLIERS / 20X,1HA,14X,1HB,14X,1HC,
     1 14X,1HD,/6H X-DIR4E15.6/ 6H Y-DIR4E15.6/ 6H Z-DIR4E15.6/ )
 2010 FORMAT(1H1,1H ,
     1 '30X40H FIXED END FORCES IN LOCAL COORDINATES '
     2//'53H TYPE   NODE       FORCE X     FORCE Y     FORCE Z  ',
     3                     '35H MOMENT X    MOMENT Y    MOMENT Z  '       )
 2011 FORMAT(1H ,I3,6X,1HI,3X,6F12.3/1H ,9X,1HJ,3X,6F12.3/)
 2013 FORMAT(1H0/
     1  60H SECTION PROPERTIES OTHER THAN SHEAR AREAS MAY NOT BE SPECIF
     2  34HIED AS ZERO. EXECUTION TERMINATED.)
 3000 FORMAT (10I5,2I6,I8)
 4000 FORMAT (22H13/D BEAM ELEMENT DATA, /// 3X,4HBEAM,3(3X,4HNODE),3X,
     1        8HMATERIAL,3X,7HSECTION,3X,17HELEMENT END LOADS,3X,
     2        9HEND CODES, / 7H NUMBER,5X,2H-I,5X,2H-J,5X,2H-K,1X,
     3        2(4X,6HNUMBER),4X,1HA,4X,1HB,4X,1HC,4X,1HD,4X,2H-I,4X,
     4        2H-J, / 1X)
 4001 FORMAT (4(2X,I5),6X,I5,5X,I5,4I5,2I6)
 4002 FORMAT (9H0BEAM NO ,I5, 26H     K NODE ON BEAM X-AXIS     ,
     .  26H......EXECUTION TERMINATED )
 4003 FORMAT(36H0ELEMENT CARD ERROR, ELEMENT NUMBER= I6)
 4004 FORMAT(1H ,31HNODAL POINT NUMBERS FOR ELEMENT,I5,'36HARE IDENTCAL
     1 EXECUTION TERMINATED.')
 4005 FORMAT(8H0ELEMENT,I5,39H HAS ZERO LENGTH. EXECUTION TERMINATED.)
      END
cMMMMMMMMMMMMM axisymmetric element (should be deleted later)

      SUBROUTINE ELAW (NUMTC,EE,E,C,P,ALP)
C
C     CALLS?  POSINV
C     CALLED BY?  PLNAX
C
      COMMON /JUNK/ MAT,NT,TEMP,REFT,BETA,TAU(4),D(4,4),CC(4,4)
     1             ,XX(4),IFILL1(342)
      COMMON /ELPAR/ NPAR(14),IFILL2(10)
      DIMENSION     E(NUMTC,11,1),EE(10),C(4,4),P(4),ALP(4)
C
C          STRESS-STRAIN LAW IN  N-S-T  SYSTEM
C
```

```
        IF (NT.NE.1)  GO TO 220
      DO 210 KK=1,10
  210 EE(KK)=E(1,KK+1,MAT)
      GO TO 260
  220 DO 230 I=2,NT
      T1=E(I-1,1,MAT)
      T2=E(I,1,MAT)
      IF(T2.GE.TEMP) GO TO 240
  230 CONTINUE
  240 CONTINUE
      RI=(T2-TEMP)/(T2-T1)
      RJ=(TEMP-T1)/(T2-T1)
      DO 250 KK=1,10
  250 EE(KK)=E(I-1,KK+1,MAT)*RI+E(I,KK+1,MAT)*RJ
  260 CONTINUE
      DO 265 II=1,4
      DO 265 KK=1,4
      C(II,KK)=0.
  265 D(II,KK)=0.
C
      C(1,1)= 1.0/ EE(1)
      C(2,2)= 1.0/ EE(2)
      C(3,3)= 1.0/ EE(3)
      C(1,2)= -EE(4)/EE(2)
      C(1,3)= -EE(5)/EE(3)
      C(2,3)= -EE(6)/EE(3)
      C(2,1)= C(1,2)
      C(3,1)= C(1,3)
      C(3,2)= C(2,3)
      C(4,4)= 1.0/ EE(7)
C
      DO 270 M=1,3
      ALP(M)  = EE(M+7)
  270 CONTINUE
      ALP(4)  = 0.0
C
C       ROTATE MATERIAL PROPERTIES TO R-Z-T SYSTEM
C
      IF(BETA.EQ.0.0) GO TO 500
      ANG=BETA/57.2957795
      SS=SIN(ANG)
      ACC=COS(ANG)
      S2=SS*SS
      C2=ACC*ACC
      SC=SS*ACC
C     SET  D  FOR  SIG(O)=D*SIG(G)
      D(1,1)=C2
      D(1,2)=S2
      D(1,4)=2.*SC
      D(2,1)=S2
      D(2,2)=C2
      D(2,4)=-D(1,4)
      D(3,3)=1.0
      D(4,1)=-SC
      D(4,2)=-D(4,1)
```

```
      D(4,4)=C2-S2
C
C     FORM  (D)TRANSPOSE * (C)
C
      DO 300 I=1,4
      DO 300 J=1,4
      SUM=0.
      DO 280 M=1,4
  280 SUM=SUM+D(M,I)*C(M,J)
  300 CC(I,J)=SUM
C
C     FORM  (D)TRANSPOSE * (C) * (D)
C
      DO 350 I=1,4
      DO 350 J=I,4
      SUM=0.
      DO 330 M=1,4
  330 SUM=SUM+CC(I,M)*D(M,J)
      C(I,J)=SUM
  350 C(J,I)=SUM
C
C     TRANSFORM THERMAL EXPANSION COEFFICIENTS
C
      XX(1)=C2*ALP(1)+S2*ALP(2)
      XX(2)=S2*ALP(1)+C2*ALP(2)
      XX(3)=ALP(3)
      XX(4)=2.*SC*(ALP(1)-ALP(2))
      DO 430 I=1,4
  430 ALP(I) = XX(I)
C
C     INVERT THE STRAIN-STRESS LAW
C
  500 CALL POSINV (C,4,4)
C
C     MODIFY FOR THE CONDITION OF PLANE STRESS
C
      IF(NPAR(5).NE.2) GO TO 660
C
      C(1,1)= C(1,1)- C(3,1)* C(1,3)/C(3,3)
      C(1,2)= C(1,2)- C(3,2)* C(1,3)/C(3,3)
      C(1,4)= C(1,4)- C(3,4)* C(1,3)/C(3,3)
      C(2,2)= C(2,2)- C(3,2)* C(2,3)/C(3,3)
      C(2,4)= C(2,4)- C(3,4)* C(2,3)/C(3,3)
      C(4,4)= C(4,4)- C(3,4)* C(4,3)/C(3,3)
C
      DO 650 I=1,4
      DO 600 J=I,4
  600 C(J,I)=C(I,J)
      C(I,3)=0.
  650 C(3,I)=0.
C
C     RESTRAINED THERMAL STRESSES
C
  660 DO 670 I=1,4
      P(I) = 0.
```

```
      DO 670 M=1,4
  670 P(I)=P(I)+C(I,M)*ALP(M)
C
  700 RETURN
      END
      SUBROUTINE CROSS(A,B,C)
C
C     CALLED BY?  PLNAX
C
      DIMENSION A(4),B(4),C(4)
      X=A(2)*B(3)-A(3)*B(2)
      Y=A(3)*B(1)-A(1)*B(3)
      Z=A(1)*B(2)-A(2)*B(1)
      C(4)=SQRT(X*X+Y*Y+Z*Z)
      C(3)=Z/C(4)
      C(2)=Y/C(4)
      C(1)=X/C(4)
      RETURN
      END
      SUBROUTINE FORMB(S,T,B)
C
C     CALLED BY?  QUAD
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/     LM(12),U(12,12),P(12,4),XM(12),
     1 TI(20,4),IX(4),IE(5),NS,D(4,4),EMUL(4,5),RR(4),ZZ(4),H(6),HS(6),
     2 HT(6),HR(6),HZ(6),FAC,XMM,PRESS,    EE(10),TTI(4),PP(12,4),THICK
     3 ,TMP(4),QP(12),ALP(4),IFILL2(4236)
      DIMENSION B(20,12)
      DIMENSION II(6),JJ(6)
      DATA II/1,2,3,4,9,10/,JJ/5,6,7,8,11,12/
C
      SM=1.0-S
      SP=1.0+S
      TM=1.0-T
      TP=1.0+T
C
      H(1)=SM*TM/4.
      H(2)=SP*TM/4.
      H(3)=SP*TP/4.
      H(4)=SM*TP/4.
      H(5)=(1.0-S*S)
      H(6)=(1.0-T*T)
C
      HS(1)=-TM/4.
      HS(2)=-HS(1)
      HS(3)=TP/4.
      HS(4)=-HS(3)
      HS(5)=-2.*S
      HS(6)=0.0
C
      HT(1)=-SM/4.
      HT(2)=-SP/4.
      HT(3)=-HT(2)
      HT(4)=-HT(1)
```

```
      HT(5)=0.0
      HT(6)=-2.*T
C
      PZT=HT(1)*ZZ(1)+HT(2)*ZZ(2)+HT(3)*ZZ(3)+HT(4)*ZZ(4)
      PZS=HS(1)*ZZ(1)+HS(2)*ZZ(2)+HS(3)*ZZ(3)+HS(4)*ZZ(4)
      PRS=HS(1)*RR(1)+HS(2)*RR(2)+HS(3)*RR(3)+HS(4)*RR(4)
      PRT=HT(1)*RR(1)+HT(2)*RR(2)+HT(3)*RR(3)+HT(4)*RR(4)
      XJ=PRS*PZT-PRT*PZS
C
      PSR=PZT/XJ
      PTR=-PZS/XJ
      PSZ=-PRT/XJ
      PTZ=PRS/XJ
C
      DO 100 I=1,6
      HR(I)=PSR*HS(I)+PTR*HT(I)
  100 HZ(I)=PSZ*HS(I)+PTZ*HT(I)
      R=H(1)*RR(1)+H(2)*RR(2)+H(3)*RR(3)+H(4)*RR(4)
      IF(NPAR(5).NE.0) R=THICK
C
C     FORM STRAIN DISPLACEMENT MATRIX
C
      DO 200 K=1,6
      I=II(K)
      J=JJ(K)
      B(1,I)=HR(K)
      B(2,J)=HZ(K)
C
C     TEST FOR HOOP STRAIN EVALUATION (AXISYMMETRIC SOLID)
C
      IF(NPAR(5).GT.0) GO TO 190
C        SET HOOP STRAIN .EQ. RADIAL STRAIN IF ON C/L AXIS
      IF(R.LT.1.0E-6)
     *B(3,I)=B(1,I)
C
      IF(R.GT.1.0E-6)
     *B(3,I)=H(K)/R
C
  190 CONTINUE
      B(4,I)=HZ(K)
  200 B(4,J)=HR(K)
C
      FAC=XJ*R
      RETURN
      END
      SUBROUTINE PLNAX (ID,X,Y,Z,T,NTC,WT,RO,WANG,E,NUMTC,NUMNP,B,BB)
C
C     CALLS?  ELAW,QUAD,VECTOR,CROSS,DOT,CALBAN
C     CALLED BY?  PLANE
C
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,1),NTC(1),WT(1),RO(1),WANG(1),
     1          E(NUMTC,11,1),T(1),B(20,12),BB(20,12)
      COMMON /ELPAR/ NPAR(14),NUMNN,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/    LM(12),S(12,12),P(12,4),XM(12),
     1 TI(20,4),IX(4),IE(5),NS,D(4,4),EMUL(4,5),RR(4),ZZ(4),H(6),HS(6),
```

```
     2 HT(6),HR(6),HZ(6),FAC,XMM,PRESS,    EE(10),TTI(4),PP(12,4),THICK
     3 ,TMP(4),TP(12),ALP(4),IFILL2(4236)
      COMMON /JUNK/ MAT,NT,TEMP,REFT,BETA,U(4),V(4),W(4),G(4),IFLL(390)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      NUME=NPAR(2)
      NUMMAT=NPAR(3)
      numee=nume
         neqq=neq
      WRITE (6,2000)  (NPAR(M),M=2,6)
C
C     READ AND PRINT OF MATERIAL PROPERTIES
C
      DO 60 M=1,NUMMAT
      READ  (5,1010)  MAT,NTC(MAT),WT(MAT),RO(MAT),WANG(MAT)
        IF  (NTC(MAT).EQ.0)   NTC(MAT)=1
      WRITE  (6,2020)  MAT,NTC(MAT),WT(MAT),RO(MAT),WANG(MAT)
      NT=NTC(MAT)
      READ  (5,1005)  ((E(I,J,MAT),J=1,11),I=1,NT)
      WRITE  (6,2010)  ((E(I,J,MAT),J=1,11),I=1,NT)
   60 CONTINUE
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.0) GO TO 75
      DO 70 M=1,NUMMAT
      WRITE  (NT8)  M,NTC(M),WT(M),WANG(M)
      NT = NTC(M)
      WRITE  (NT8)  ((E(I,J,M),J=1,11),I=1,NT)
   70 CONTINUE
   75 CONTINUE
C
C     ELEMENT LOAD CASE MULTIPLIERS
C
      READ  (5,1002)   ((EMUL(I,J),J=1,5),I=1,4)
      WRITE  (6,2004)  ((EMUL(I,J),J=1,5),I=1,4)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE  (NT8)  ((EMUL(I,J),J=1,5),I=1,4)
C
C       READ AND PRINT OF ELEMENT PROPERTIES
C
      WRITE  (6,2002)
      N=0
  130 READ(5,1003)  M,(IE(I),I=1,5),REFT,PRESS,NS,KG,THICK
      MAT=IE(5)
      IF(KG.EQ.0)  KG=1
      IF  (NPAR(5).EQ.1)  THICK=1.0
      IF(NS.EQ.0)  NS=4
      IF(NS.LT.4)  NS=1
      IF(  (IE(3).EQ.IE(4)).AND.(NS.EQ.20)  )  NS=16
  140 N=N+1
      IF(M.EQ.N)  GO TO 145
      DO 142 I=1,4
  142 IX(I)=IX(I)+KG
```

```
      GO TO 149
  145 DO 148 I=1,4
  148 IX(I)=IE(I)
C
C        FORM CONSTITUTIVE LAW AND COMPUTE THERMAL STRESSES
C
  149 NT=NTC(MAT)
      WRITE (6,2003) N,IX,MAT,REFT,PRESS,NS,KG,THICK
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.0) GO TO 150
      WRITE (NT8) N,IX,MAT,REFT,PRESS,NS,THICK
      GO TO 153
  150 CONTINUE
      I=IX(1)
      J=IX(2)
      K=IX(3)
      L=IX(4)
      TEMP = (T(I)+T(J)+T(K)+T(L))/4.0
      BETA=WANG(MAT)
      XMM=RO(MAT)
      WGT=WT(MAT)
      CALL ELAW (NUMTC,EE,E,D,TTI,ALP)
C
C     CALCULATE ELEMENT STIFFNESS MATRIX
C
  153 IF(NPAR(1).EQ.3) GO TO 160
      ND=8
      DO 155 I=1,4
      II=IX(I)
      RR(I)=Y(II)
      ZZ(I)=Z(II)
      TMP(I) = T(II)
      LM(I)=ID(II,2)
  155 LM(I+4)=ID(II,3)
      IF(MODEX.EQ.1) GO TO 300
C
      CALL QUAD (B,BB)
C
      DO 158 I=1,4
      DO 157 L=1,4
      P(I,L)=P(I,L)+XM(I)*WGT*EMUL(L,4)
  157 P(I+4,L)=P(I+4,L)+XM(I)*WGT*EMUL(L,5)
      XM(I)=XM(I)*XMM
  158 XM(I+4)=XM(I)
      GO TO 300
C
  160 ND = 12
      IF(MODEX.EQ.1) GO TO 165
      CALL VECTOR(V,X(I),Y(I),Z(I),X(J),Y(J),Z(J))
      CALL VECTOR(G,X(I),Y(I),Z(I),X(L),Y(L),Z(L))
      CALL CROSS(V,G,W)
      CALL CROSS(W,V,U)
      CALL VECTOR(W,X(I),Y(I),Z(I),X(K),Y(K),Z(K))
      RR(1)=0.0
      ZZ(1)=0.0
```

```
      RR(2)=V(4)
      ZZ(2)=0.0
      RR(3)=W(4)*DOT(W,V)
      ZZ(3)=W(4)*DOT(W,U)
      RR(4)=G(4)*DOT(G,V)
      ZZ(4)=G(4)*DOT(G,U)
C
  165 DO 170 I=1,4
      II=IX(I)
      TMP(I) = T(II)
      LM(I)=ID(II,1)
      LM(I+4)=ID(II,2)
  170 LM(I+8)=ID(II,3)
      IF(MODEX.EQ.1) GO TO 300
C
      CALL QUAD (B,BB)
C
      DO 190 I=1,3
      DO 190 K=1,4
      KK=4*(I-1)+K
      DO 180 L=1,4
  180 PP(KK,L)=V(I)*P(K,L)+U(I)*P(K+4,L)
      DO 190 J=1,3
      DO 190 L=1,4
      LL=4*(J-1)+L
  190 BB(KK,LL)=V(I)*(S(K,L)*V(J)+S(K,L+4)*U(J))
     1 +U(I)*(S(K+4,L)*V(J)+S(K+4,L+4)*U(J))
C
      DO 196 I=1,12
      DO 194 L=1,4
  194 P(I,L)=PP(I,L)
      DO 196 J=1,12
      S(I,J)=BB(I,J)
  196 S(J,I)=S(I,J)
C
      DO 210 K=1,NS
      DO 200 L=1,4
      DO 200 J=1,3
      LL=4*(J-1)+L
  200 BB(K,LL)=B(K,L)*V(J)+B(K,L+4)*U(J)
      DO 210 J=1,12
  210 B(K,J)=BB(K,J)
C
      DO 220 I=1,4
      DO 215 L=1,4
      P(I   ,L)=P(I   ,L)+XM(I)*WGT*EMUL(L,3)
      P(I+4,L)=P(I+4,L)+XM(I)*WGT*EMUL(L,4)
  215 P(I+8,L)=P(I+8,L)+XM(I)*WGT*EMUL(L,5)
      XM(I)=XM(I)*XMM
      XM(I+4)=XM(I)
  220 XM(I+8)=XM(I)
C
C     CALCULATION OF BAND WIDTH AND WRITES ELEMENT MATRICES ON TAPES
C
  300 CALL CALBAN (MBAND,NDIF,LM,XM,S,P,ND,12,NS)
```

```
      IF(MODEX.EQ.1) GO TO 310
      WRITE (1) ND,NS,(LM(I),I=1,ND),(( B(I,J),I=1,NS),J=1,ND),
     1 ((TI(I,J),I=1,NS),J=1,4)
  310 IF(N.EQ.NUME) RETURN
      IF(N.EQ.M) GO TO 130
      GO TO 140
C
 1002 FORMAT (5F10.0)
 1003 FORMAT (6I5,2F10.0,2I5,F10.0)
 1005 FORMAT (8F10.0/3F10.0)
 1010 FORMAT (2I5,3F10.0)
 2000 FORMAT (// 23H NUMBER OF ELEMENTS   =, I6 /
     1           23H NUMBER OF MATERIALS  =, I6 /
     2           23H MAXIMUM TEMPERATURES  ,    /
     3           23H PER MATERIAL         =, I6 /
     4           23H ANALYSIS CODE        =, I6 /
     5           23H CODE FOR INCLUSION    ,    /
     6           23H OF BENDING MODES     =, I6 /
     7           23H   EQ.0, INCLUDE       ,    /
     8           23H   GT.0, SUPPRESS      ,    //// 1X)
 2002 FORMAT (8H1ELEMENT,26X,4HMATL,5X,9HREFERENCE,3X,8HI-J FACE,3X,
     1        6HSTRESS, / 2X,6HNUMBER,5X,1HI,5X,1HJ,5X,1HK,5X,1HL,2X,
     2        4HTYPE,3X,11HTEMPERATURE,3X,8HPRESSURE,3X,6HOPTION,4X,
     3        2HKG,3X,9HTHICKNESS, / 1X)
 2003 FORMAT (I8,5I6,F14.3,E11.3,I9,I6,F12.4)
 2004 FORMAT (/// 25H ELEMENT LOAD MULTIPLIERS, // 10H LOAD CASE,4X,
     1        11HTEMPERATURE,3X,8HPRESSURE,3X,9HX-GRAVITY,3X,
     2        9HY-GRAVITY,3X,9HZ-GRAVITY, // 5X,1HA,F19.3,F11.3,3F12.3 /
     3        5X,1HB,F19.3,F11.3,3F12.3 /    5X,1HC,F19.3,F11.3,3F12.3 /
     4        5X,1HD,F19.3,F11.3,3F12.3  )
 2010 FORMAT (F12.2,3E12.4,3F9.4,E12.4,3E14.4)
 2020 FORMAT (/// 25H MATERIAL I.D. NUMBER   =, I5 /
     1            25H NUMBER OF TEMPERATURES =, I5 /
     2            25H WEIGHT DENSITY         =, E14.4 /
     3            25H MASS   DENSITY         =, E14.4 /
     4            25H BETA ANGLE             =, F9.3 //
     5        12H TEMPERATURE,8X,4HE(N),8X,4HE(S),8X,4HE(T),3X,6HNU(NS),
     6        3X,6HNU(NT),3X,6HNU(ST),7X,5HG(NS),6X,8HALPHA(N),6X,
     7        8HALPHA(S),6X,8HALPHA(T)   )
      END
      SUBROUTINE QUAD (B,BB)
C
C     CALLS?  FORMB,VECTOR
C     CALLED BY?  PLNAX
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/    LM(12),S(12,12),P(12,4),XM(12),
     1 TI(20,4),IX(4),IE(5),NS,D(4,4),EMUL(4,5),RR(4),ZZ(4),H(6),HS(6),
     2 HT(6),HR(6),HZ(6),FAC,XMM,PRESS,    EE(10),TTI(4),PP(12,4),THICK
     3 ,TMP(4),TP(12),ALP(4),IFILL2(4236)
      COMMON /JUNK/ MAT,NT,TEMP,REFT,BETA,IFILL1(422)
      DIMENSION B(20,12),BB(20,12)
      DIMENSION SS(2),TT(2),HH(2),SSS(5),TTT(5),IVECT(4),JVECT(4),V(4)
      DATA SSS/0.,-1.,1.,0.,0./, TTT/0.,0.,0.,-1.,1./
      DATA SS/-0.57735026918963,0.57735026918963/
```

```
      DATA TT/-0.57735026918963,0.57735026918963/
      DATA HH/1.,1./, IVECT/4,2,1,3/, JVECT/1,3,2,4/
C
      DO 170 J=1,12
      XM(J)=0.0
      TP(J) = 0.0
      DO 160 I=1,20
      BB(I,J)=0.0
  160 B(I,J)=0.0
      DO 170 I=1,12
  170 S(I,J)=0.0
C
      DO 500 II=1,2
      DO 500 JJ=1,2
      CALL FORMB(SS(II),SS(JJ),B)
      TEMP = 0.0
      DO 200 I=1,4
  200 TEMP = TEMP + H(I)* TMP(I)
      FAC=FAC*HH(JJ)*HH(II)
      FTP = TEMP - REFT
      DO 400 J=1,12
      D1=(D(1,1)*B(1,J)+D(1,2)*B(2,J)+D(1,3)*B(3,J)+D(1,4)*B(4,J))*FAC
      D2=(D(2,1)*B(1,J)+D(2,2)*B(2,J)+D(2,3)*B(3,J)+D(2,4)*B(4,J))*FAC
      D3=(D(3,1)*B(1,J)+D(3,2)*B(2,J)+D(3,3)*B(3,J)+D(3,4)*B(4,J))*FAC
      D4=(D(4,1)*B(1,J)+D(4,2)*B(2,J)+D(4,3)*B(3,J)+D(4,4)*B(4,J))*FAC
      TP(J) = TP(J) + FTP*(D1*ALP(1) +D2*ALP(2) + D3*ALP(3) + D4*ALP(4))
      DO 400 I=J,12
      S(I,J)=S(I,J)+B(1,I)*D1+B(2,I)*D2+B(3,I)*D3+B(4,I)*D4
  400 S(J,I)=S(I,J)
      DO 450 I=1,4
  450 XM(I)=XM(I)+FAC*H(I)
  500 CONTINUE
C
C     FORM STRESS DISDLACEMENT MATRIX
C
      LL=NS/4
      DO 530 L=1,LL
      CALL FORMB(SSS(L),TTT(L),BB)
C
      TEMP = 0.0
      DO 515 K=1,4
  515 TEMP = TEMP + H(K)* TMP(K)
      FAC = TEMP - REFT
      DO 530 II=1,4
      I=II+4*(L-1)
      TI(I,4) = -TTI(II)* FAC
      DO 530 J=1,12
      B(I,J)=0.0
      DO 530 K=1,4
  530 B(I,J)=B(I,J)+D(II,K)*BB(K,J)
C
C     ELIMINATE EXTRA DEGREES OF FREEDOM
C
      IF( IX(3).EQ.IX(4) )  GO TO 560
      IF( NPAR(6).NE.0   )  GO TO 560
```

```
      DO 550 NN=1,4
      L=12-NN
      K=L+1
      C = TP(K)/S(K,K)
      DO 535 J=1,NS
  535 TI(J,4) = TI(J,4) + C* B(J,K)
      DO 550 I=1,L
      C=S(I,K)/S(K,K)
      TP(I) = TP(I) - C* TP(K)
      DO 540 J=1,NS
  540 B(J,I)=B(J,I)-C*B(J,K)
      DO 550 J=1,L
  550 S(I,J)=S(I,J)-C*S(K,J)
C
C     ROTATE STRESS-DISPLACEMENT TRANSFORMATION TO GIVE STRESSES
C     NORMAL AND PARALLEL TO SIDES.  SIMILARLY, ROTATE INITIAL STRESSES.
C
  560 NSET = LL-1
      IF( NSET.LE.0 )  GO TO 730
      DO 720 L=1,NSET
      IV = IVECT(L)
      JV = JVECT(L)
      CALL VECTOR (V,RR(IV),ZZ(IV),0.0,RR(JV),ZZ(JV),0.0)
      S2 = V(1)*V(1)
      C2 = V(2)*V(2)
      SC =-V(1)*V(2)
      I1 = 4*L+1
      I2 = I1+1
      I4 = I1+3
      T1 = TI(I1,4)
      T2 = TI(I2,4)
      T4 = TI(I4,4)
      T5 = 2.0*SC*T4
      TI(I1,4) = C2*T1+S2*T2+T5
      TI(I2,4) = S2*T1+C2*T2-T5
      TI(I4,4) = SC*(T2-T1)+(C2-S2)*T4
      DO 710 J=1,8
      B1 = B(I1,J)
      B2 = B(I2,J)
      B4 = B(I4,J)
      B5 = 2.0*SC*B4
      B(I1,J) = C2*B1+S2*B2+B5
      B(I2,J) = S2*B1+C2*B2-B5
  710 B(I4,J) = SC*(B2-B1)+(C2-S2)*B4
  720 CONTINUE
  730 CONTINUE
C
      DO 660 L=1,4
      DO 600 I=1,NS
  600 TI(I,L) = TI(I,4)* EMUL(L,1)
      DO 660 I=1,8
  660 P(I,L) = TP(I)* EMUL(L,1)
C
C     CALCULATE PRESSURE LOADS ON I-J FACE
C
```

```
      DR=RR(2)-RR(1)
      DZ=ZZ(1)-ZZ(2)
      RI=PRESS*(2.*RR(1)+RR(2))/6.
      RJ=PRESS*(2.*RR(2)+RR(1))/6.
      IF(NPAR(5).EQ.0) GO TO 670
      RI=PRESS*THICK/2.
      RJ=RI
  670 DO 700 L=1,4
      P(1,L)=P(1,L)+DZ*RI*EMUL(L,2)
      P(5,L)=P(5,L)+DR*RI*EMUL(L,2)
      P(2,L)=P(2,L)+DZ*RJ*EMUL(L,2)
  700 P(6,L)=P(6,L)+DR*RJ*EMUL(L,2)
      RETURN
      END
      SUBROUTINE VECTOR(V,XI,YI,ZI,XJ,YJ,ZJ)
C
C     CALLED BY?  PLNAX,QUAD
C
      DIMENSION V(4)
      X=XJ-XI
      Y=YJ-YI
      Z=ZJ-ZI
      V(4)=SQRT(X*X+Y*Y+Z*Z)
C
      V(3)=Z/V(4)
      V(2)=Y/V(4)
      V(1)=X/V(4)
      RETURN
      END
      SUBROUTINE POSINV(A,NMAX,NDD)
C
C     CALLED BY?  ELAW
C
      DIMENSION A(NDD,NDD)
C
      DO 200 N=1,NMAX
C
      D=A(N,N)
      DO 100 J=1,NMAX
  IF(D.EQ.0.)D=0.005
  100 A(N,J)=-A(N,J)/D
C
      DO 150 I=1,NMAX
      IF(N-I) 110,150,110
  110 DO 140 J=1,NMAX
      IF(N-J) 120,140,120
  120 A(I,J)=A(I,J)+A(I,N)*A(N,J)
  140 CONTINUE
  150 A(I,N)=A(I,N)/D
C
      A(N,N)=1.0/D
C
  200 CONTINUE
C
      RETURN
```

```
      END
      FUNCTION DOT(A,B)
C
C     CALLED BY?  PLNAX
C
      DIMENSION A(4),B(4)
      DOT=A(1)*B(1)+A(2)*B(2)+A(3)*B(3)
      RETURN
      END
      SUBROUTINE PLANE
C
C     CALLS?  PLNAX,STRSC
C     CALLED BY?  ELTYPE
C
      COMMON /one/ A(1)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/ NS,ND,B(42,63),TI(42,4),LM(63)
      COMMON /JUNK/  LT,LH,L,IPAD,SG(20),SIG(7),EXTRA(150),N6,N7,N8,N9,
     1          N10,N11,N12,IFILL(65)
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      DIMENSION STRLAB(5)
      DATA STRLAB/3HCEN,3HL-I,3HJ-K,3HI-J,3HK-L/
C

      IF(NPAR(1).EQ.0) GO TO 200
      IF(NPAR(1).EQ.3) NPAR(5)=2
      IF(NPAR(5).EQ.0) WRITE (6,2000)
      IF(NPAR(5).EQ.1) WRITE (6,2001)
      IF(NPAR(5).EQ.2) WRITE (6,2002)
      IF  (NPAR(1).EQ.3) WRITE (6,2003)
      IF  (NPAR(6).NE.0) WRITE (6,2004)
      IF(NPAR(3).EQ.0) NPAR(3)=1
      IF(NPAR(4).EQ.0) NPAR(4)=1
      N6=N5+NUMNP
      N7=N6+NPAR(3)
      N8=N7+NPAR(3)
      N9=N8+NPAR(3)
      N10=N9+NPAR(3)
      N11=N10+11*NPAR(4)*NPAR(3)
      N12=N11+240
      MM=N12+240-MTOT
      IF(MM.GT.0) CALL ERROR(MM)
C
      CALL PLNAX(A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),
     1           A(N9),A(N10),NPAR(4),NUMNP,A(N11),A(N12))
C
      RETURN
C
  200 WRITE (6,2006)
      NUME=NPAR(2)
      DO 800 MM=1,NUME
      CALL STRSC(A(N1),A(N3),NEQ,0)
C***  STRESS PORTHOLE
```

```
      IF(N10SV.EQ.1)
     *WRITE (NT10) NS
      IF(NS.EQ.1) GO TO 800
      WRITE (6,3000) MM
      DO 700 L=LT,LH
      CALL STRSC(A(N1),A(N3),NEQ,1)
      ITAG = 0
  510 DO 600 KK=1,NS,4
      ITAG = ITAG + 1
      DO 520 I=1,4
      II=KK-1+I
  520 SIG(I)=SG(II)
      CC=(SIG(1)+SIG(2))/2.0
      BB=(SIG(1)-SIG(2))/2.
      CR=SQRT(BB**2+SIG(4)**2)
      SIG(5)=CC+CR
      SIG(6)=CC-CR
      SIG(7)=0.0
      IF((BB.EQ.0.0) .AND. (SIG(4).EQ.0.0)) GO TO 530
      SIG(7)=28.648*ATAN2(SIG(4),BB)
C***  STRESS PORTHOLE
  530 IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,(SIG(I),I=1,7)
  600 WRITE (6,3001) L,STRLAB(ITAG),(SIG(I),I=1,7)
      WRITE (6,3002)
  700 CONTINUE
  800 CONTINUE
      RETURN
 2000 FORMAT (22H1AXISYMMETRIC ANALYSIS )
 2001 FORMAT (22H1PLANE STRAIN ANALYSIS )
 2002 FORMAT (22H1PLANE STRESS ANALYSIS )
 2003 FORMAT (18H MEMBRANE ELEMENTS )
 2004 FORMAT (30H INCOMPATIBLE MODES SUPPRESSED )
 2006 FORMAT (54H1T W O - D I M E N S I O N A L   F I N I T E   E L E M,
     1        8H E N T S,/// 8X,32H1.  CENTROID STRESSES REFERENCED,
     2       26H TO LOCAL Y-Z COORDINATES.,/  8X, 12H2.  MID-SIDE,
     3       51H STRESSES ARE NORMAL AND PARALLEL TO ELEMENT EDGES.,
     4       // 1X)
 3000 FORMAT (10HOELEMENT (,I5,1H),// 2X,4HLOAD,2X,3HLOC,12X,3HS11,12X,
     1       3HS22,12X,3HS33,12X,3HS12,10X,5HS-MAX,10X,5HS-MIN,5X,
     2       5HANGLE, / 1X)
 3001 FORMAT (I6,2X,A3,6E15.5,F10.2)
 3002 FORMAT (1H0)
      END
       subroutine assm(a,b,ll,ntr,neq)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      COMMON /EM/ LRD,ND,LM(63),IPAD,SS(2331)
        dimension a(ntr),b(neq,ll)
        dimension sd(24,24)
       neq=neqq
       nume=numee
       do 71 i=1,nterms
71     a(i)=0
c      contribution from each element
```

```
      rewind 14
      rewind 13
      do 72 i=1,nume
      read(14) lrd,nd,(lm(ii),ii=1,nd)
      read(13) ((sd(ii,jj),jj=1,nd),ii=1,nd)
      do 79 j=1,nd
      jj=lm(j)
      if(jj.eq.0) go to 79
      do 76 k=1,nd
      kk=lm(k)
      if(kk.eq.0.or.kk.lt.jj) go to 76
      locate=naxa(jj)+kk-jj
      a(locate)=a(locate)+sd(j,k)
76    continue
79    continue
72    continue
      return
      end

      SUBROUTINE ADDSTF (A,B,STR,TMASS,NUMEL,NBLOCK,NE2B,LL,MBAND,ANORM,
     1NVV)
C
C
C     CALLED BY?  MAIN
C
C     FORMS GLOBAL EQUILIBRIUM EQUATIONS IN BLOCKS
C
      DIMENSION A(NE2B,MBAND),B(NE2B,LL),STR(4,LL),TMASS(NE2B)
C
      COMMON /DYN/   NT,NOT,ALFA,DT,BETA,NFN,NGM,NAT,NDYN
      COMMON /EM/ LRD,ND,LM(63),IPAD,SS(2331)
      COMMON /EXTRA/ MODEX,NT8,IFILL(14)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      NEQB=NE2B/2
      K=NEQB+1
      X=NBLOCK
      MB=SQRT(X)
      MB=MB/2+1
      NEBB=MB*NE2B
      MM=1
      NDEG=0
      NVV=0
      ANORM=0.
      NSHIFT=0
      REWIND 3
      REWIND 4
      REWIND 9
C
C     READ ELEMENT LOAD MULTIPLIERS
C
      WRITE (6,2000)
      DO 50 L=1,LL
      READ  (5,1002)   (STR(I,L),I=1,4)
```

```
   50 WRITE (6,2002) L,(STR(I,L),I=1,4)
      IF(MODEX.EQ.0) WRITE (8) STR
C
C     FOR A STEP-BY-STEP ANALYSIS  (NDYN.EQ.4)  READ THE SOLUTION
C     CONTROL CARD.  THE TIME STEP (DT) AND THE DAMPING COEFFICIENTS
C     (ALFA/BETA) ARE REQUIRED FOR THE ASSEMBLY OF THE EFFECTIVE
C     SYSTEM STIFFNESS MATRIX IN THIS ROUTINE.
C
      IF(NDYN.NE.4) GO TO 65
C
      READ  (5,1004) NFN,NGM,NAT,NT,NOT,DT,ALFA,BETA
      WRITE (6,2004) NFN,NGM,NAT,NT,NOT,DT,ALFA,BETA
      IF(NAT.EQ.0) NAT = 1
      IF(NOT.EQ.0) NOT = 1
      IF(DT.GT.1.0E-12) GO TO 55
      WRITE (6,3000)
      STOP
C
C     COMPUTE INTEGRATION COEFFICIENTS FOR ASSEMBLY OF EFFECTIVE
C     SYSTEM STIFFNESS (STEP-BY-STEP ANALYSIS ONLY)
C
   55 TETA = 1.4
      DT1  = TETA*DT
      DT2  = DT1**2
      AO   = (6.+3.*ALFA*DT1)/(DT2+3.*BETA*DT1)
C
   65 IF(MODEX.EQ.1) RETURN
C
C     FORM EQUATIONS IN BLOCKS     ( 2 BLOCKS AT A TIME)
C
      DO 1000 M=1,NBLOCK ,2
      DO 100 I=1,NE2B
      DO 100 J=1,MBAND
  100 A(I,J)=0.
      READ (3)  ((B(I,L),I=1,NEQB),L=1,LL),(TMASS(I),I=1,NEQB)
      IF (M.EQ.NBLOCK) GO TO 200
      READ (3)  ((B(I,L),I=K,NE2B),L=1,LL),(TMASS(I),I=K,NE2B)
  200 CONTINUE
C
      REWIND 7
      REWIND 2
      NA=7
      NUME=NUM7
      IF (MM.NE.1) GO TO 75
      NA=2
      NUME=NUMEL
      NUM7 =0
C
   75 DO 700 N=1,NUME
      READ (NA) LRD,ND,(LM(I),I=1,ND),(SS(I),I=1,LRD)
      MSHFT = ND * (ND+1)/2 +4 *ND
      DO 600 I=1,ND
      LMN=1-LM(I)
      II=LM(I)-NSHIFT
      IF (II.LE.0.OR.II.GT.NE2B) GO TO 600
```

```
      IMS=I+MSHFT
      TMASS(II)=TMASS(II)+ SS(IMS)
      DO 300 L=1,LL
      DO 300 J=1,4
      KK = ND *(ND+1)/2 + ND*(J-1)
  300 B(II,L)=B(II,L)+SS(I+KK)*STR(J,L)
      DO 500 J=1,ND
      JJ=LM(J)+LMN
      IF(JJ) 500,500,390
  390 IF(J-I) 396,394,394
  394 KK = ND*I-(I-1)*I/2 +J-ND
      GO TO 400
  396 KK =ND*J -(J-1)*J/2+I-ND
  400 A(II,JJ)=A(II,JJ)+SS(  KK)
  500 CONTINUE
  600 CONTINUE
C
C     DETERMINE IF STIFFNESS IS TO BE PLACED ON TAPE 7
C
      IF (MM.GT.1) GO TO 700
      DO 650 I=1,ND
      II=LM(I) -NSHIFT
      IF(II.GT.NE2B.AND.II.LE.NEBB) GO TO 660
  650 CONTINUE
      GO TO 700
  660 WRITE (7) LRD,ND,(LM(I),I=1,ND),(SS(I),I=1,LRD)
      NUM7=NUM7+1
C
  700 CONTINUE
      DO 710 L=1,NEQB
      ANORM=ANORM + A(L,1)
      IF (A(L,1).NE.O.) NDEG=NDEG + 1
      IF (A(L,1).EQ.O.) A(L,1)=1.E+20
      IF (TMASS(L).NE.O.) NVV=NVV + 1
  710 CONTINUE
C
C     FOR STEP-BY-STEP ANALYSIS ADD THE MASS CONTRIBUTIONS TO
C     THE EQUATION DIAGONAL COEFFICIENTS
C
      IF(NDYN.NE.4) GO TO 716
      DO 714 I=1,NEQB
  714 A(I,1) = A(I,1) + AO* TMASS(I)
      WRITE (4) ((A(I,J),I=1,NEQB),J=1,MBAND)
      GO TO 718
  716 WRITE (4) ((A(I,J),I=1,NEQB),J=1,MBAND),((B(I,L),I=1,NEQB),L=1,LL)
cmo
  718 WRITE (9)  (TMASS(I),I=1,NEQB)
C

c      moayyad
c         do 212 i=1,neqb
c212      write(6,213) (a(i,j),j=1,mband)
c213      format(6e12.5)
       IF(M.EQ.NBLOCK) GO TO 1000
       DO 720 L=K,NE2B
```

```
      ANORM=ANORM + A(L,1)
      IF  (A(L,1).NE.0.) NDEG=NDEG + 1
      IF  (A(L,1).EQ.0.) A(L,1)=1.E+20
      IF  (TMASS(L).NE.0.) NVV=NVV + 1
  720 CONTINUE
C
      IF(NDYN.NE.4) GO TO 726
      DO 724 I=K,NE2B
  724 A(I,1) = A(I,1) + AO* TMASS(I)
      WRITE (4)  ((A(I,J),I=K,NE2B),J=1,MBAND)
      go to 728
726   write(4)((a(i,j),i=k,ne2b),j=1,mband),((b(i,l),i=k,ne2b),l=1,11)
  728 WRITE (9)   (TMASS(I),I=K,NE2B)
C
      IF (MM.EQ.MB) MM=0
      MM=MM+1
 1000 NSHIFT=NSHIFT+NE2B
      IF (NDEG.GT.0) GO TO 730
      WRITE (6,1010)
      STOP
  730 ANORM=(ANORM/NDEG)*1.E-8
C
      RETURN
 1002 FORMAT (4F10.0)
 1004 FORMAT (5I5,3F10.0)
 1010 FORMAT (51HOSTRUCTURE WITH NO DEGREES OF FREEDOM  CHECK DATA   )
 2000 FORMAT (/// 10H STRUCTURE,13X,7HELEMENT,4X,4HLOAD,4X,
     1 11HMULTIPLIERS,/ 10H LOAD CASE,12X,1HA,9X,1HB,9X,1HC,9X,1HD,/ 1X)
 2002 FORMAT (I6,7X,4F10.3)
 2004 FORMAT (45H1S T E P - B Y - S T E P   S O L U T I O N   ,
     1         37HC O N T R O L   I N F O R M A T I O N, ///
     2 5X, 35HNUMBER OF TIME VARYING FUNCTIONS  =, I5    //
     3 5X, 35HGROUND MOTION INDICATOR           =, I5    /
     4 8X, 10HEQ.0, NONE, /
     5 8X, 29HGT.0, READ ACCELERATION INPUT, //
     6 5X, 35HNUMBER OF ARRIVAL TIMES           =, I5    /
     7 8X, 26HEQ.0, ALL FUNCTIONS ARRIVE, /
     8 8X, 18H       AT TIME ZERO, //
     9 5X, 35HNUMBER OF SOLUTION TIME STEPS     =, I5    //
     A 5X, 35HOUTPUT (PRINT) INTERVAL           =, I5    //
     B 5X, 35HSOLUTION TIME INCREMENT           =, E14.4 //
     C 5X, 30HMASS-     PROPORTIONAL DAMPING, /
     D 5X, 35HCOEFFICIENT (ALPHA)               =, E14.4 //
     E 5X, 30HSTIFFNESS-PROPORTIONAL DAMPING, /
     F 5X, 35HCOEFFICIENT (BETA)                =, E14.4 /// 1X)
 3000 FORMAT (27HO*** ERROR   ZERO TIME STEP, / 1X)
      END
c********************************* s7.frc
      SUBROUTINE ADDMAS (TMASS,BLKMAS,NEQ,NEQB,NBLOCK)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE READS THE SYSTEM MASS MATRIX IN BLOCKED FORM
C     FROM *TAPE9* AND ASSEMBLES THE BLOCKS INTO A SINGLE VECTOR
C     *NEQ* WORDS IN LENGTH -- I.E., SYSTEM MASS MATRIX (DIAGONAL)
```

```
C     IS STORED IN CORE.  SYSTEM MASS MATRIX *TMASS* IS SAVED ON
C     *TAPE3*.
C
      DIMENSION      TMASS(NEQ),BLKMAS(NEQB)
C
      NT3 = 3
      REWIND NT3
      NT9 = 9
      REWIND NT9
C
      KSHIFT = 0
C
C     LOOP ON THE TOTAL NUMBER OF SYSTEM EQUATION BLOCKS
C
      DO 200 K=1,NBLOCK
      READ (NT9) BLKMAS
      K1 = KSHIFT
      DO 100 L=1,NEQB
      K1 = K1+1
      IF(K1.GT.NEQ) GO TO 250
      TMASS(K1) = BLKMAS(L)
  100 CONTINUE
      KSHIFT = KSHIFT+NEQB
  200 CONTINUE
C
  250 WRITE (NT3) TMASS
C
      RETURN
      END
       SUBROUTINE BANDET (A,B,V,MAXA,NN,NWA,RA,NSCH,DET,ISCALE,KK)
C
C     CALLED BY?  SECNTD
C
       COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
       DIMENSION A(NWA),B(1),V(1),MAXA(1)
C
       NR=NN-1
       IF (KK-2) 100,700,800
C
 100   TOL=1.0E+07
       RTOL=1.0E-10
C     SCALE=2.0D0**200
C***      SCALE=2.D0**166
      SCALE=2.D0**10
       NTF=3
       IS=1
 120   REWIND NSTIF
       READ (NSTIF) A
       DO 140 I=1,NN
 140   A(I)=A(I)-RA*B(I)
 160   IF (NWA.EQ.NN) GO TO 230
       DO 200 N=1,NR
       IH=N+NWA-NN
 210   IF (A(IH)) 220,215,220
 215   IH=IH-NN
```

```
      GO TO 210
 220  MAXA(N)=IH
      PIV=A(N)
      IF(PIV) 221,500,221
  500 IS = IS+1
      IF(IS.LE.NTF) GO TO 502
  501 WRITE (6,1000) NTF,RA
      STOP
  502 RA = RA*(1.0-RTOL)
      GO TO 120
 221  IL=N+NN
      L=N
      DO 240 I=IL,IH,NN
      L=L+1
      C=A(I)
      IF (C) 225,240,225
 225  C=C/PIV
      IF (ABS(C) .LT. TOL) GO TO 235
 226  IS=IS+1
      IF (IS.LE.NTF) GO TO 245
      GO TO 501
 245  RA=RA*(1.0-RTOL)
      GO TO 120
 235  J=L-I
      DO 260 K=I,IH,NN
 260  A(K+J)=A(K+J)-C*A(K)
      A(I)=C
 240  CONTINUE
 200  CONTINUE
 230  IF (A(NN).NE.0.0) GO TO 280
      AA=ABS(A(1))
      DO 290 I=2,NR
 290  AA=AA+ABS(A(I))
      A(NN)=-(AA/NR)*1.0E-16
C
 280  NSCH=0
      ISC=0
      DET=1.0
      DO 300 I=1,NN
      IF (ABS(DET) .LT. SCALE) GO TO 320
      DET=DET/SCALE
      ISC=ISC+1
 320  DET=DET*A(I)
 300  IF (A(I).LT.0.) NSCH=NSCH+1
C
      IF (ISCALE.LT.1000) GO TO 340
      ISCALE=ISC
      GO TO 900
 340  IF (ISC-ISCALE) 350,900,370
 350  DET=DET/SCALE
      GO TO 900
 370  DET=DET*SCALE
      GO TO 900
C
 700  IL=NN
```

```
      DO 400 N=1,NR
      C=V(N)
      V(N)=C/A(N)
      IF (NWA-NN) 410,400,410
 410  IL=IL+1
      IH=MAXA(N)
      K=N
      DO 420 I=IL,IH,NN
      K=K+1
 420  V(K)=V(K)-C*A(I)
 400  CONTINUE
      V(NN)=V(NN)/A(NN)
C
 800  IF (NWA-NN) 430,900,430
 430  N=NN
      DO 440 L=2,NN
      N=N-1
      IL=N+NN
      IH=MAXA(N)
      K=N
      DO 460 I=IL,IH,NN
      K=K+1
 460  V(N)=V(N)-A(I)*V(K)
 440  CONTINUE
 900  RETURN
C
 1000 FORMAT (37HO***ERROR   SOLUTION STOP IN *BANDET*, / 12X,
     1        1H(,I3,37H) TRIANGULAR FACTORIZATIONS ATTEMPTED, / 12X,
     2        16HCURRENT SHIFT = ,E20.14 / 1X)
C
      END
      SUBROUTINE BENDDC (NEL,NI,NJ,X1,X2,X3,R,KODE,A,MODEX,THETA,TOL,PI)
C
C     CALLED BY?  PIPEK
C
C     COMPUTATION OF DIRECTION COSINE ARRAY FOR THE LOCAL AXES OF A
C     CIRCULAR BEND PIPE ELEMENT
C
C     NEL           =  ELEMENT NUMBER
C     NI            =  NODE NUMBER AT END I
C     NJ            =  NODE NUMBER AT END J
C     X1            =  GLOBAL COORDINATES OF END I
C     X2            =  GLOBAL COORDINATES OF END J
C     X3            =  GLOBAL COORDINATES OF THE THIRD POINT
C     KODE          =  CODE DEFINING THE THIRD POINT
C                      (EQ.1, TANGENT INTERSECTION POINT)
C                      (EQ.2, CENTER OF CURVATURE)
C     R             =  RADIUS OF THE BEND
C     A             =  MATRIX OF DIRECTION COSINES RELATING LOCAL TO THE
C                      GLOBAL SYSTEM.  A(I,J) IS THE PROJECTION ON THE
C                      I-TH GLOBAL AXIS OF A UNIT VECTOR IN THE LOCAL
C                      J-DIRECTION.
C     MODEX         =  EXECUTION MODE
C                      (EQ.0, SOLUTION)
C                      (EQ.1, DATA CHECK)
```

```
C     THETA          =  CENTRAL ANGLE SUBTENDED BY THE ARC OF THE BEND
C     TOL            =  DIMENSIONAL TOLERANCE USED FOR ERROR TESTING
C     PI             =  3.14159...
C
      DIMENSION X1(3),X2(3),X3(3),A(3,3),B(3)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      GO TO (10,110),KODE
C
C     TANGENT INTERSECTION IS THE THIRD POINT
C
C        1. LOCAL X-AXIS VECTOR
C
   10 A(1,1) = X3(1)-X1(1)
      A(2,1) = X3(2)-X1(2)
      A(3,1) = X3(3)-X1(3)
      XL1T = A(1,1)**2 + A(2,1)**2 + A(3,1)**2
      XL1T =SQRT(XL1T)
      IF(XL1T.GT.1.0E-8) GO TO 20
      NN = NI
   15 WRITE (6,3000) NEL,NN
      MODEX = 1
      RETURN
   20 DUM = 1.0/ XL1T
      DO 25 K=1,3
   25 A(K,1) = A(K,1)* DUM
C
C        2. VECTOR FROM TANGENT POINT TO NODE J
C
      DO 30 K=1,3
   30 B(K) = X2(K)-X3(K)
      XLT2 = B(1)**2 + B(2)**2 + B(3)**2
      XLT2 =SQRT(XLT2)
      IF(XLT2.GT.1.0E-8) GO TO 40
      NN = NJ
      GO TO 15
C
C        3. COMPARE DISTANCES BETWEEN THE NODES AND THE COMMON TANGENT
C           INTERSECTION POINT
C
   40 DIF =ABS(XL1T-XLT2)
      IF(DIF.LE.TOL) GO TO 42
      WRITE (6,3010) NEL,TOL,XL1T,XLT2
      MODEX = 1
      RETURN
C
   42 CONTINUE
C
C        4. LOCAL Z-AXIS
C
      A(1,3) = A(2,1)*B(3) - A(3,1)*B(2)
      A(2,3) = A(3,1)*B(1) - A(1,1)*B(3)
      A(3,3) = A(1,1)*B(2) - A(2,1)*B(1)
      DUM = 0.0
```

```
      DO 44 K=1,3
   44 DUM = DUM + A(K,3)**2
      DUM =SQRT(DUM)
      IF(DUM.GT.1.0E-8) GO TO 46
      WRITE (6,3060) NEL
      MODEX = 1
      RETURN
   46 DUM = 1.0/DUM
      DO 48 K=1,3
   48 A(K,3) = A(K,3) * DUM
C
C        5. LOCAL Y-AXIS
C
      A(1,2) = A(2,3)*A(3,1) - A(3,3)*A(2,1)
      A(2,2) = A(3,3)*A(1,1) - A(1,3)*A(3,1)
      A(3,2) = A(1,3)*A(2,1) - A(2,3)*A(1,1)
C
C        6. COMPUTE THE CENTRAL ANGLE
C
      DUM = XL1T/R
      THETA = 2.0D0*ATAN(DUM)
   50 CONTINUE
      IF(THETA.GT.1.0E-8 .AND. THETA.LE.PI) RETURN
      DUM = THETA*180.0/PI
      WRITE (6,3020) DUM,NEL
      MODEX = 1
      RETURN
C
C     CENTER OF CURVATURE IS THE THIRD POINT
C
C        1. LOCAL Y-AXIS VECTOR
C
  110 A(1,2) = X3(1)-X1(1)
      A(2,2) = X3(2)-X1(2)
      A(3,2) = X3(3)-X1(3)
      D1C = 0.0
      DO 120 K=1,3
  120 D1C = D1C + A(K,2)**2
      D1C =SQRT(D1C)
      IF(D1C.GT.1.0E-8) GO TO 130
      NN = NI
  125 WRITE (6,3030) NEL,NN
      MODEX = 1
      RETURN
  130 DUM = 1.0/ D1C
      DO 135 K=1,3
  135 A(K,2) = A(K,2)* DUM
C
C        2. COMPUTE THE VECTOR FROM NODE J TO THE C.C.
C
      B(1) = X3(1)-X2(1)
      B(2) = X3(2)-X2(2)
      B(3) = X3(3)-X2(3)
      D2C = 0.0
      DO 140 K=1,3
```

```
  140 D2C = D2C + B(K)**2
      D2C =SQRT(D2C)
      IF(D2C.GT.1.0E-8) GO TO 150
      NN = NJ
      GO TO 125
  150 CONTINUE
C
C        3. COMPARE COMPUTED RADII VERSUS THE INPUT VALUE
C
      DIF =ABS(R-D1C)
      IF(DIF.LT.TOL) GO TO 165
      NN = NI
      RR = D1C
  160 WRITE (6,3040) NN,NEL,R,RR
      MODEX = 1
      RETURN
  165 DIF =ABS(R-D2C)
      IF(DIF.LT.TOL) GO TO 170
      NN = NJ
      RR = D2C
      GO TO 160
C
C        4. LOCAL Z-AXIS VECTOR
C
  170 A(1,3) = A(2,2)*B(3) - A(3,2)*B(2)
      A(2,3) = A(3,2)*B(1) - A(1,2)*B(3)
      A(3,3) = A(1,2)*B(2) - A(2,2)*B(1)
      DUM = 0.0
      DO 172 K=1,3
  172 DUM = DUM + A(K,3)**2
      DUM =SQRT(DUM)
      IF(DUM.LT.1.0E-8) GO TO 177
      DUM = 1.0/DUM
      DO 173 K=1,3
  173 A(K,3) = A(K,3) * DUM
C
C        5. TEST FOR NODES I AND J COINCIDENT
C
      CHORD = 0.0
      DO 175 K=1,3
  175 CHORD = CHORD + (X2(K)-X1(K))**2
      CHORD =SQRT(CHORD)
      IF(CHORD.GT.1.0E-8) GO TO 180
  177 WRITE (6,3050) NI,NJ,NEL
      MODEX = 1
      RETURN
C
C        6. LOCAL X-AXIS VECTOR
C
  180 A(1,1) = A(2,2)*A(3,3) - A(3,2)*A(2,3)
      A(2,1) = A(3,2)*A(1,3) - A(1,2)*A(3,3)
      A(3,1) = A(1,2)*A(2,3) - A(2,2)*A(1,3)
C
C        7. COMPUTE THE CENTRAL ANGLE
C
```

```
      DUM = 0.5*CHORD/R
C***      THETA = 2.0D0*DARSIN(DUM)
        THETA = 2.0D0*ASIN(DUM)
      GO TO 50
C
 3000 FORMAT (25HOERROR***  BEND ELEMENT (,I4,19H) HAS ZERO DISTANCE,
     1 15H BETWEEN NODE (,I4,31H) AND THE TANGENT INTERSECTION., / 1X)
 3010 FORMAT (45HOERROR***  TANGENT LENGTHS FOR BEND ELEMENT (,I4,
     1 27H) ARE NOT EQUAL TO WITHIN (,E11.4, 2H)., /
     2 11X,23HI-NODE TANGENT LENGTH =,E20.8, /
     3 11X,23HJ-NODE TANGENT LENGTH =,E20.8, / 1X)
 3020 FORMAT (30HOERROR***  THE CENTRAL ANGLE (,F8.3,10H) FOR BEND,
     1 10H ELEMENT (,I4,18H) IS OUT OF RANGE., / 11X,
     2 38HTHETA MUST BE GT.0 AND LT.180 DEGREES., / 1X)
 3030 FORMAT (25HOERROR***  BEND ELEMENT (,I4,19H) HAS ZERO DISTANCE,
     1 15H BETWEEN NODE (,I4,30H) AND THE CENTER OF CURVATURE.,/ 1X)
 3040 FORMAT (36HOERROR***  COMPUTED RADIUS TO NODE (,I4,10H) FOR BEND,
     1 10H ELEMENT (,I4,38H) IS DISCREPANT FROM THE INPUT RADUIS., /
     2 11X, 17HRADIUS INPUT    =,E20.8 /
     3 11X, 17HRADIUS COMPUTED =,E20.8 / 1X)
 3050 FORMAT (44HOERROR***  ZERO CHORD LENGTH BETWEEN NODES (,I4,
     1 7H) AND (,I4,19H) IN BEND ELEMENT (,I4,2H)., / 1X)
 3060 FORMAT (51HOERROR***  TANGENT INTERSECTION POINT FOR ELEMENT (,
     1         I4,18H) IS ON THE CHORD., / 1X)
C
      END
C
C     CALLS?  PINVER
C     CALLED BY?  PIPEK
C
C     COMPUTATION OF THE ELEMENT STIFFNESS AND LOAD MATRICES FOR A
C     CIRCULARLY CURVED PIPE BEND ELEMENT.
C
C
C     ALFAV          =  SHAPE FACTOR FOR SHEAR DISTORTION
C                       (GT.99.99, NEGLECT)
C     NOAX           =  CODE FOR NEGLECTING AXIAL DEFORMATIONS
C                       (EQ.1, NEGLECT)
C     E              =  YOUNG*S MODULUS
C     XNU            =  POISSON*S RATIO
C     XKP            =  PRESSURE DEPENDENT FLEXIBILITY FACTOR
C     AREA           =  SECTION AREA
C     XMI            =  MOMENT OF INERTIA
C     T              =  ANGLE OF THE BEND, THETA
C     ST             =  SIN(THETA)
C     CT             =  COS(THETA)
C     NODE           =  NODE NUMBER AT END J OF THE BEND
C     NEL            =  PIPE ELEMENT NUMBER
C     MODEX          =  EXECUTION MODE
C                       (EQ.1, DATA CHECK)
C     F(6,6)         =  FLEXIBILITY MATRIX AT NODE J
C     R              =  RADIUS OF THE BEND
C     THERM          =  THERMAL EXPANSION COEFFICIENT
C     P              =  INTERNAL PIPE PRESSURE
C     WALL           =  PIPE WALL THICKNESS
```

```
C      DOUT           =  OUTSIDE DIAMETER OF THE PIPE
C      B              =  FREE END DEFLECTIONS AT NODE J DUE TO
C                        (1)  UNIFORM LOAD IN THE X(I) DIRECTION
C                        (2)  UNIFORM LOAD IN THE Y(I) DIRECTION
C                        (3)  UNIFORM LOAD IN THE Z(I) DIRECTION
C                        (4)  UNIFORM THERMAL EXPANSION (DT=1)
C                        (5)  P,    INTERNAL PRESSURE
C      H              =  FORCE TRANSFORMATION RELATING REACTIONS AT NODE I
C                        DUE TO UNIT LOADS AT NODE J
C      S              =  LOCAL BEND ELEMENT STIFFNESS MATRIX
C      FEF            =  FIXED END FORCES (ACTING ON THE NODES) DUE TO
C                        (1)  UNIFORM LOAD IN THE X(I) DIRECTION
C                        (2)  UNIFORM LOAD IN THE Y(I) DIRECTION
C                        (3)  UNIFORM LOAD IN THE Z(I) DIRECTION
C                        (4)  UNIFORM THERMAL EXPANSION (DT=1)
C                        (5)  P,    INTERNAL PRESSURE
C      XM             =  LUMPED MASS MATRIX
C      SA             =  STRESS-DISPLACEMENT TRANSFORMATION RELATING THE
C                        12 GLOBAL COMPONENTS OF DISPLACEMENT TO THE 6
C                        LOCAL COMPONENTS OF MEMBER LOADS LOCATED AT NODE
C                        I, MIDPOINT OF THE ARC AND AT NODE J.
C      FEFC           =  FIXED-END FORCE CORRECTIONS TO THE MEMBER LOADS
C                        DUE TO THE FIVE (5) TYPES OF ELEMENT LOADS
C      XMAS           =  MASS   PER UNIT LENGTH OF THE SECTION
C      DC             =  ARRAY OF DIRECTION COSINES WHICH TRANSFORMS LOCAL
C                        VECTORS TO GLOBAL VECTORS
       SUBROUTINE BENDKS
       COMMON /PIPEC/ALFAV,E,XNU,XKP,T,NOAX,NODE,NEL,
      1              MODEX,R,THERM,P,AREA,XMI,WALL,DOUT,XMAS
       COMMON /EM/    IXX(14),S(12,12),RF(12,4),XM(12),SA(18,12),
      1               SF(18,4),FEF(12,5),FEFC(18,5),F(6,6),B(6,6),
      2               H(6,6),DC(3,3),IFILL2(3606)
       COMMON /ELPAR/ NPAR(14),IFILL1(10)
       common /say/ neqq,numee,loopur,nnblock,nterms,option
       common /what/ naxa(10000),irowl(10000),icolh(10000)
C
       DIMENSION      COL(6)
C
C      SET THE FACTOR FOR AXIAL DEFORMATIONS
C
       AXIAL = 1.0
       IF(NOAX.EQ.1) AXIAL = 0.0
C
C      SET THE FACTOR FOR SHEAR DEFORMATIONS (EQ.0, NEGLECT)
C
       XKAP = ALFAV
       IF(ALFAV.GT.99.99) XKAP = 0.0
C
C      SET THE FLEXIBILITY FACTORS
C
       XKO = XKP
       XK1 = XKP
C
C      COMPUTE THE MATERIAL FACTORS
C
```

```
      RE = 1.0/E
      XNU1 = 1.0+XNU
C
C     COMPUTE SECTION PROPERTY CONSTANTS
C
      RA = AXIAL*R*RE/AREA
      RV = XKAP*XNU1*R*RE/AREA
      RT = 0.5*XNU1*R*RE/XMI
      RBO = 0.5*XKO*R*RE/XMI
      RB1 = XK1*R*RE/XMI
      R2 = R**2
C
C     COMPUTE COMMON TRIGONOMETRIC CONSTANTS
C
      ST =SIN(T)
      CT =COS(T)
      S2T =SIN(2.0*T)
      C2T =COS(2.0*T)
      T2 = T**2
C
C     FORM THE NODE FLEXIBILITY MATRIX AT NODE J REFERENCED TO THE
C     LOCAL (X,Y,Z) COORDINATE SYSTEM AT NODE I.
C
C     X - DIRECTION...  IN-PLANE TANGENT TO THE BEND AT NODE I AND
C                       DIRECTED TOWARD NODE J
C     Y - DIRECTION...  IN-PLANE AND DIRECTED RADIALLY INWARD TO THE
C                       CENTER OF CURVATURE
C     Z - DIRECTION...  OUT OF PLANE AND ORTHOGONAL TO X AND Y
C
      DO 50 I=1,6
      DO 50 K=1,6
      F(I,K) = 0.0
   50 CONTINUE
C
C     A X I A L
C
      F(1,1) = F(1,1) + 0.25*RA*(2.0*T+S2T)
      F(2,2) = F(2,2) + 0.25*RA*(2.0*T-S2T)
C     N O T E   (COEFFICIENT CHANGE)
      F(1,2) = F(1,2) + 0.50*RA* ST**2
C
C     S H E A R
C
      F(1,1) = F(1,1) + 0.5*RV*(2.0*T-S2T)
      F(2,2) = F(2,2) + 0.5*RV*(2.0*T+S2T)
      F(3,3) = F(3,3) + 2.0*RV* T
C     N O T E   (SIGN CHANGE)
      F(1,2) = F(1,2) -     RV* ST**2
C
C     T O R S I O N
C
      F(3,3) = F(3,3) + 0.5*RT*R2*(6.0*T+S2T-8.0*ST)
      F(4,4) = F(4,4) + 0.5*RT*    (2.0*T+S2T)
      F(5,5) = F(5,5) + 0.5*RT*    (2.0*T-S2T)
      F(3,4) = F(3,4) +      RT*R *(ST-T*CT)
```

```
      F(3,5)  = F(3,5)  +      RT*R  *(2.0-2.0*CT-T*ST)
      F(4,5)  = F(4,5)  + 0.5*RT*     (1.0-C2T)
C
C     B E N D I N G
C
      F(1,1)  = F(1,1)  + 0.25*RB1*R2*(2.0*T*(2.0+C2T)-3.0*S2T)
      F(2,2)  = F(2,2)  + 0.25*RB1*R2*(2.0*T*(2.0-C2T)+3.0*S2T-8.0*ST)
      F(3,3)  = F(3,3)  + 0.50*RB0*R2*(2.0*T-S2T)
      F(4,4)  = F(4,4)  + 0.50*RB0*    (2.0*T-S2T)
      F(5,5)  = F(5,5)  + 0.50*RB0*    (2.0*T+S2T)
      F(6,6)  = F(6,6)  +       RB1*T
      F(1,2)  = F(1,2)  + 0.25*RB1*R2*(1.0+3.0*C2T+2.0*T*S2T-4.0*CT)
      F(1,6)  = F(1,6)  -       RB1*R *(ST-T*CT)
      F(2,6)  = F(2,6)  +       RB1*R *(T*ST+CT-1.0)
      F(3,4)  = F(3,4)  +       RB0*R *(ST-T*CT)
      F(3,5)  = F(3,5)  -       RB0*R *T*ST
      F(4,5)  = F(4,5)  - 0.50*RB0*    (1.0-C2T)
C
      DO 60 I=1,6
      DO 60 K=1,6
      F(K,I)  = F(I,K)
   60 CONTINUE
C**** PRINT THE NODE FLEXIBILITY MATRIX
      IF(NPAR(10).LT.1) GO TO 6700
      WRITE (6,4000)
      WRITE (6,4010) ((F(I,K),K=1,6),I=1,6)
 6700 CONTINUE
C****
C
C     FORM THE NODE STIFFNESS MATRIX
C
      CALL PINVER (F,6,6,NODE,NEL,MODEX)
C**** PRINT THE NODE STIFFNESS MATRIX
      IF(NPAR(10).LT.1) GO TO 6701
      WRITE (6,4020)
      WRITE (6,4030) ((F(I,K),K=1,6),I=1,6)
 6701 CONTINUE
C****
C
C     COMPUTE THE DEFLECTIONS/ROTATIONS (MEASURED IN THE X,Y,Z SYSTEM
C     AT NODE I) AT NODE J DUE TO UNIFORM LOADS IN EACH OF THE X,Y,Z
C     DIRECTIONS (AT I).  THE UNIFORM LOADS ARE DIRECTION INVARIANT
C     WITH POSITION ALONG THE ARC, AND NODE I IS FIXED WHILE NODE J IS
C     COMPLETELY FREE.
C
      DO 70 I=1,6
      DO 70 K=1,3
      B(I,K)  = 0.0
   70 CONTINUE
C
C     A X I A L
C
      RA = 0.125*RA*R
      B(1,1)  = B(1,1)  +      RA*(2.0*T2-C2T+1.0)
      B(2,2)  = B(2,2)  +      RA*(2.0*T2+C2T-1.0)
```

```
C     N O T E   (COEFFICIENT CHANGE)
      B(1,2) = B(1,2) +     RA*(2.0*T-S2T)
C     N O T E   (COEFFICIENT CHANGE)
      B(2,1) = B(2,1) +     RA*(2.0*T-S2T)
C
C     S H E A R
C
      RV = 0.25*RV*R
      B(1,1) = B(1,1) +     RV*(2.0*T2+C2T-1.0)
      B(2,2) = B(2,2) +     RV*(2.0*T2-C2T+1.0)
      B(3,3) = B(3,3) + 4.0*RV*T2
C     N O T E   (SIGN CHANGE)
      B(1,2) = B(1,2) -     RV*(2.0*T-S2T)
C     N O T E   (SIGN CHANGE)
      B(2,1) = B(2,1) -     RV*(2.0*T-S2T)
C
C     T O R S I O N
C
      RT = RT*R2
      B(3,3) = B(3,3) + 0.5*RT*R*(1.0+2.0*T2-4.0*T*ST-C2T)
      B(4,3) = B(4,3) +     RT*  (2.0-2.0*CT-T*ST)
      B(5,3) = B(5,3) +     RT*  (T*(2.0+CT)-3.0*ST)
C
C     B E N D I N G
C
      RB0 = RB0*R2
      RB1 = RB1*R2
      B(1,1) = B(1,1) + 0.125*RB1*R*(7.0+2.0*T2+9.0*C2T+4.0*T*S2T
     1                              -16.0*CT)
      B(2,2) = B(2,2) + 0.125*RB1*R*(1.0+2.0*T2-9.0*C2T-4.0*T*S2T
     1                              +8.0*(CT-T*ST))
      B(3,3) = B(3,3) + 0.500*RB0*R*(3.0+C2T-4.0*CT)
      B(1,2) = B(1,2) + 0.125*RB1*R*(9.0*S2T-4.0*T*(C2T+2.0*CT)-6.0*T)
      B(2,1) = B(2,1) + 0.125*RB1*R*(9.0*S2T-4.0*T*C2T-24.0*ST+10.0*T)
      B(4,3) = B(4,3) +       RB0*  (2.0-2.0*CT-T*ST)
      B(5,3) = B(5,3) -       RB0*  (ST-T*CT)
      B(6,1) = B(6,1) -       RB1*  (2.0-2.0*CT-T*ST)
      B(6,2) = B(6,2) +       RB1*  (2.0*ST-T-T*CT)
C
C     COMPUTE THE FREE NODE DEFLECTIONS AT END J DUE TO A UNIFORM
C     THERMAL EXPANSION
C
      DO 80 I=1,6
      B(I,4) = 0.0
   80 CONTINUE
C
      DUM = R*THERM
      B(1,4) = DUM*ST
      B(2,4) = DUM*(1.0-CT)
C
C     COMPUTE THE FREE NODE DEFLECTIONS AT END J DUE TO PRESSURE
C
      DO 90 I=1,6
      B(I,5) = 0.0
   90 CONTINUE
```

```
C
C      COMPUTE THE ANGLE CHANGE AND END DISPLACEMENTS AT THE FREE END
C      OF THE BEND DUE TO INTERNAL PRESSURE, P.
C
       RM = (DOUT-WALL)*0.5
       KK = 1
       GO TO (92,94),KK
C
C      MEL REPORT 10-66, EQUATION (3-29).
C
   92 CONTINUE
       DUM = 3.14159265*RM**4*P*T
       DUM = 0.5*DUM*RE/XMI
       DU2 = RM/R
       BTA = DUM*(2.0-2.0*XNU + (3.0+1.5*XNU)*DU2**2)
       GO TO 96
C
C      C. S. PARKER, EQUATION (10), 2-28-69.
C
   94 CONTINUE
       DU2 = R/RM
       DUM = P*RM*0.5*RE/WALL
       DU3 = 1.0 + DUM*(1.0-XNU*(2.0*DU2-1.0)/(DU2-1.0))
       BTA = DU3/(1.0 + DUM*(2.0-XNU))
       BTA = T*(1.0-BTA)
C
   96 CONTINUE
       DUM = R/T*BTA
       B(1,5) = DUM*(ST-T*CT)
       B(2,5) = DUM*(1.0-CT-T*ST)
       B(6,5) =-BTA
C
C      AXIAL GROWTH DUE TO PRESSURE.  MEL REPORT 10-66, EQUATION (3-28).
C
       DUM = 0.5* P* RM* RE* (1.0-2.0*XNU)* R/ WALL
       B(1,5) = B(1,5) + DUM* ST
       B(2,5) = B(2,5) + DUM* (1.0-CT)
C**** PRINT THE FREE END DEFLECTIONS
       IF(NPAR(10).LT.1) GO TO 6702
       WRITE (6,4050)
       WRITE (6,4060) ((B(I,K),K=1,5),I=1,6)
 6702 CONTINUE
C****
C
C      SET UP THE FORCE TRANSFORMATION RELATING REACTIONS AT NODE I
C      ACTING ON THE MEMBER END DUE TO UNIT LOADS APPLIED TO THE MEMBER
C      END AT NODE J.
C
       DO 100 I=1,6
       DO 100 K=1,6
       H(I,K) = 0.0
  100 CONTINUE
C
       DO 105 K=1,6
       H(K,K) =-1.0
```

```
  105 CONTINUE
C
      H(4,3)  =-R*(1.0-CT)
      H(5,3)  = R* ST
      H(6,1)  =-H(4,3)
      H(6,2)  =-H(5,3)
C
C     FORM THE UPPER TRIANGULAR PORTION OF THE LOCAL ELEMENT STIFFNESS
C     MATRIX FOR THE BEND
C
      DO 110 K=1,6
      DO 110 I=K,6
      S(K+6,I+6)  = F(K,I)
  110 CONTINUE
C
      DO 130 IR=1,6
      DO 130 IC=1,6
      S(IR,IC+6)  = 0.0
      DO 120 IN=1,6
      S(IR,IC+6)  = S(IR,IC+6)  + H(IR,IN)* F(IN,IC)
  120 CONTINUE
  130 CONTINUE
C
      DO 150 IR=1,6
      DO 150 IC=IR,6
      S(IR,IC)  = 0.0
      DO 140 IN=1,6
      S(IR,IC)  = S(IR,IC)  + S(IR,IN+6)* H(IC,IN)
  140 CONTINUE
  150 CONTINUE
C
C     REFLECT FOR SYMMETRY
C
      DO 160 I=1,12
      DO 160 K=1,12
      S(K,I)  = S(I,K)
  160 CONTINUE
C**** PRINT THE BEND LOCAL STIFFNESS MATRIX
      IF(NPAR(10).LT.1) GO TO 6703
      WRITE (6,4500)
      WRITE (6,4510)  ((S(I,J),J=1,6 ),I=1,12)
      WRITE (6,4510)  ((S(I,J),J=7,12),I=1,12)
 6703 CONTINUE
C****
C
C     COMPUTE THE RESTRAINED NODE FORCES ACTING ON THE NODES OF THE
C     BEND DUE TO THE MEMBER LOADINGS
C
      DO 180 I=1,5
      DO 180 J=1,12
      FEF(J,I)  = 0.0
      DO 170 K=1,6
      FEF(J,I)  = FEF(J,I)  - S(J,K+6)* B(K,I)
  170 CONTINUE
  180 CONTINUE
```

```
C
C     FOR THE DISTRIBUTED LOADS SUPERIMPOSE THE CANTILEVER REACTIONS
C     ACTING ON THE ELEMENT AT NODE I.
C
      FEF(1,1) = FEF(1,1) - R*T
      FEF(6,1) = FEF(6,1) + R2*(T-ST)
C
      FEF(2,2) = FEF(2,2) - R*T
      FEF(6,2) = FEF(6,2) - R2*(1.0-CT)
C
      FEF(3,3) = FEF(3,3) - R*T
      FEF(4,3) = FEF(4,3) - R2*(T-ST)
      FEF(5,3) = FEF(5,3) + R2*(1.0-CT)
C**** PRINT THE FIXED END QUANTITIES
      IF(NPAR(10).LT.1) GO TO 6704
      WRITE (6,4600)
      WRITE (6,4610) ((FEF(I,J),J=1,5),I=1,12)
 6704 CONTINUE
C****
C
C     FORM THE LUMPED MASS MATRIX
C
      DUM = 0.5*R*T*XMAS
      DO 200 K=1,3
      XM(K)   = DUM
      XM(K+6) = DUM
      XM(K+3) = 0.0
      XM(K+9) = 0.0
  200 CONTINUE
C
C     COMPUTE THE FIXED-NODE CORRECTIONS TO THE MEMBER LOADS RESULTING
C     FROM ELEMENT LOADINGS.  FORCES ACT ON THE SEGMENT BETWEEN THE
C     POINT WHERE EVALUATED AND NODE I.
C
C        1. AT NODE I (ACTING ON NODE I)
C
      DO 210 I=1,5
      DO 210 K=1,6
      FEFC(K,I) = -FEF(K,I)
  210 CONTINUE
C
C        2. AT NODE J (ROTATE IN-PLANE FROCES AN AMOUNT THETA)
C
      DO 220 I=1,5
      DO 215 K=1,4,3
      FEFC(K+12,I) =  CT* FEF(K+6,I) + ST* FEF(K+7,I)
      FEFC(K+13,I) = -ST* FEF(K+6,I) + CT* FEF(K+7,I)
      FEFC(K+14,I) =      FEF(K+8,I)
  215 CONTINUE
  220 CONTINUE
C
C        3. AT THE MIDPOINT OF THE ARC BETWEEN NODES I AND J.
C
C           A. TRANSFER FORCES AT NODE J TO THE MIDPOINT AND ROTATE
C              AN AMOUNT THETA/2
```

```
C
      S12T =SIN(0.5*T)
      C12T =COS(0.5*T)
      DX   = R*(ST-S12T)
      DY   = R*(C12T-CT)
C
      DO 230 I=1,5
      XM10 = FEF(10,I) + FEF(9,I)* DY
      XM11 = FEF(11,I) - FEF(9,I)* DX
      FEFC( 7,I) =  C12T* FEF(7,I) + S12T* FEF(8,I)
      FEFC( 8,I) = -S12T* FEF(7,I) + C12T* FEF(8,I)
      FEFC( 9,I) =  FEF(9,I)
      FEFC(10,I) =  C12T* XM10     + S12T* XM11
      FEFC(11,I) = -S12T* XM10     + C12T* XM11
  230 FEFC(12,I) =  FEF(12,I) - FEF(7,I)* DY + FEF(8,I)* DX
C
C           B. FOR THE DISTRIBUTED LOADS SUPERIMPOSE THE RESULTANT
C              OF THE APPLIED LOADING TRANSFERRED TO THE MIDPOINT OF
C              THE ARC AND ROTATE AN AMOUNT THETA/2 (IN-PLANE)
C
      DDX = R*(2.0*(C12T-CT)/T - S12T)
      DDY = R*(2.0*(S12T-ST)/T + C12T)
      DUM = R*T*0.5
C
      FEFC( 7,1) = FEFC( 7,1) + C12T* DUM
      FEFC( 8,1) = FEFC( 8,1) - S12T* DUM
      FEFC(12,1) = FEFC(12,1) - DDY * DUM
C
      FEFC( 7,2) = FEFC( 7,2) + S12T* DUM
      FEFC( 8,2) = FEFC( 8,2) + C12T* DUM
      FEFC(12,2) = FEFC(12,2) + DDX * DUM
C
      FEFC( 9,3) = FEFC( 9,3) + DUM
      XM10 =  DDY* DUM
      XM11 = -DDX* DUM
      FEFC(10,3) = FEFC(10,3) + C12T* XM10 + S12T* XM11
      FEFC(11,3) = FEFC(11,3) - S12T* XM10 + C12T* XM11
C**** PRINT THE FIXED-END CORRECTIONS
      IF(NPAR(10).LT.1) GO TO 6705
      WRITE (6,4650)
      WRITE (6,4660) ((FEFC(I,J),J=1,5),I=1,18)
 6705 CONTINUE
C****
C
C     FORM THE TRANSFORMATION RELATING GLOBAL DISPLACEMENTS AND MEMBER
C     FORCES AT NODE I, MIDPOINT AND NODE J.
C
C        1. STRESS RESULTANTS AT NODE I
C
      DO 260 K1=1,10,3
      NRS = K1-1
      DO 260 K2=1,10,3
      NCS = K2-1
      DO 250 IR=1,3
      NR = NRS+IR
```

```
      DO 250 IC=1,3
      NC = NCS+IC
      SA(NR,NC) = 0.0
      DO 240 IN=1,3
      N = NCS+IN
      SA(NR,NC) = SA(NR,NC) - S(NR,N)* DC(IC,IN)
  240 CONTINUE
  250 CONTINUE
  260 CONTINUE
C
C        2. STRESS RESULTANTS AT NODE J
C
      H(1,1) = CT
      H(1,2) = ST
      H(2,1) =-ST
      H(2,2) = CT
      H(3,3) = 1.0
C
      DO 290 K1=7,10,3
      NRS = K1-1
      DO 280 IR=1,3
      NR = NRS+IR
      DO 280 IC=1,12
      SA(NR+6,IC) = 0.0
      DO 270 IN=1,3
      N = NRS+IN
      SA(NR+6,IC) = SA(NR+6,IC) - H(IR,IN)* SA(N,IC)
  270 CONTINUE
  280 CONTINUE
  290 CONTINUE
C
C        3. STRESS RESULTANTS AT THE MIDPOINT OF THE ARC
C
      H(1,1) =  C12T
      H(1,2) =  S12T
      H(2,1) = -S12T
      H(2,2) =  C12T
      H(3,3) =  1.0
      DO 300 I=1,3
      DO 300 K=1,3
  300 H(I+3,K+3) = H(I,K)
      H(4,3) =  DY* C12T - DX* S12T
      H(5,3) = -DY* S12T - DX* C12T
      H(6,1) = -DY
      H(6,2) =  DX
C
      DO 320 IC=1,12
      DO 310 N=1,6
  310 COL(N) = SA(N+6,IC)
      DO 320 IR=1,6
      SA(IR+6,IC) = 0.0
      DO 315 IN=1,6
      SA(IR+6,IC) = SA(IR+6,IC) - H(IR,IN)* COL(IN)
  315 CONTINUE
  320 CONTINUE
```

```
C**** PRINT THE STRESS DISPLACEMENT TRANSFORMATION
      IF(NPAR(10).LT.1) GO TO 6706
      WRITE (6,4700)
      WRITE (6,4710) ((SA(I,J),J=1, 6),I=1,18)
      WRITE (6,4710) ((SA(I,J),J=7,12),I=1,18)
 6706 CONTINUE
C****
C
 4000 FORMAT (/// 24H NODE FLEXIBILITY MATRIX, // 1X)
 4010 FORMAT ( 1X / (6E20.8) )
 4020 FORMAT (/// 22H NODE STIFFNESS MATRIX, // 1X)
 4030 FORMAT ( 1X / (6E20.8) )
 4050 FORMAT (/// 42H FREE NODE DISPLACEMENTS  (5 MEMBER LOADS), // 1X)
 4060 FORMAT (1X / (5E20.8) )
 4500 FORMAT (23H1LOCAL STIFFNESS MATRIX, // 1X)
 4510 FORMAT (// (/6E15.6) )
 4600 FORMAT (// 17H0FIXED END FORCES, // 1X)
 4610 FORMAT (5E20.8)
 4650 FORMAT (// 43H0STRESS CORRECTIONS DUE TO FIXED END FORCES, // 1X)
 4660 FORMAT (5E20.8)
 4700 FORMAT (//35H0STRESS-DISPLACEMENT TRANSFORMATION, / 1X)
 4710 FORMAT (/// (6E20.8) )
C
      RETURN
      END
      SUBROUTINE BOUND
C
C     CALLS?  CLAMP,STRSC
C     CALLED BY?  ELTYPE
C
      COMMON /one/ A(1)
c***      COMMON A(7100)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /JUNK/  LT,LH,L,IPAD,SIG(20),IFILL(386)
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      IF  (NPAR(1).EQ.0) GO TO 500
C
      CALL CLAMP  (NPAR(2),A(N1),A(N2),A(N3),A(N4),NUMNP,MBAND)
C
      RETURN
C
  500 WRITE (6,2002)
      NUME=NPAR(2)
       numee=nume
       neqq=neq
      DO 800 MM=1,NUME
      CALL STRSC  (A(N1),A(N3),NEQ,0)
      WRITE (6,2001)
      DO 800 L=LT,LH
      CALL STRSC  (A(N1),A(N3),NEQ,1)
      WRITE (6,3002) MM,L,(SIG(I),I=1,2)
C***  STRESS PORTHOLE
```

```
      IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,SIG(1),SIG(2)
  800 CONTINUE
      RETURN
C
 2001 FORMAT (/)
 2002 FORMAT (48H1B O U N D A R Y   E L E M E N T   F O R C E S /,
     1        14H M O M E N T S, // 8H ELEMENT,3X,4HLOAD,14X,5HFORCE,
     2        9X,6HMOMENT, / 8H  NUMBER,3X,4HCASE, // 1X)
 3002 FORMAT (I8,I7,4X,2E15.5)
      END
      SUBROUTINE BRICK8 (S,STR,NBRK8,NMAT,NLD,ID,X,Y,Z,T,EE,ENU,RHO,
     .                   ALPT,KTYPE,PR,YREF,NFACE,NUMNP)
C
C     CALLS?  DERIV,LOAD,LOSTR,CALBAN
C     CALLED BY?  THREED
C
C     STIFFNESS SUBROUTINE FOR 24 D.F. ISOPARAMETRIC HEXAHEDRON
C     LINEAR ELASTIC ISOTROPIC MATERIAL
C     'NINT*NINT*NINT' GAUSSIAN INTEGRATION RULE USED (NINT=1,2,3,4)
C
      DIMENSION KTYPE(1),PR(1),YREF(1),NFACE(1)
      DIMENSION T(1)
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,6)
      COMMON/EM/LM(24),ND,NS, SS(24,24),RF(24,4),XM(24),SA(12,24),
     .    SF(12,4),IFILL2(3048)
      EQUIVALENCE (IS1,SF(4)) , (IS2,SF(6))
      DIMENSION EE(1),ENU(1),RHO(1),ALPT(1)
      COMMON /GASS/ XK(4,4),WGT(4,4),IPERM(3)
      COMMON /JUNK/ E1,E2,E3,DET,MLD(4),KLD(4),MULT(4),NP(8),INP(8),
     .              A(3,3),P(3,11),B(3,3),XX(8,3),Q(11),DL(8),
     .              TT(24),XLF(4),YLF(4),ZLF(4),TLF(4),PLF(4),
     .              REFT,INEL,ININT,IMAT,IINC,TTEMP,NEL,ML,NINT,MAT,
     .              INC,IPAD,TAG,TEMP,SKIP,I,J,K,L,FAC,CC1,CC2,CC3,CC4,
     .              G,DEN,FACT,GT,GG,C1,C2,C3,C,K1,K2,IFILL1(64)
      COMMON /ELPAR/ NPAR(14),NUMN ,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      DIMENSION S(33,33),STR(12,33)
      DIMENSION E(3,3)
      DIMENSION IS(2),ISP(2)
C
      DATA STAR/'*'/,BLNK/' '/
      DIMENSION STPTS(7,3)
      DATA STPTS / 0. , 1. ,-1. , 0. , 0. , 0. , 0. ,
     .             0. , 0. , 0. , 1. ,-1. , 0. , 0. ,
     .             0. , 0. , 0. , 0. , 0. , 1. ,-1. /
      XK(1,1)=0.0D0
      XK(2,1)=0.0D0
      XK(3,1)=0.0D0
      XK(4,1)=0.0D0
      XK(1,2)=-.5773502691896D0
      XK(2,2)=-XK(1,2)
      XK(3,2)=0.0D0
```

```
      XK(4,2)=0.0D0
      XK(1,3)=-.774596669241 5D0
      XK(2,3)=0.0D0
      XK(3,3)=-XK(1,3)
      XK(4,3)=0.0D0
      XK(1,4)=-.861136311594 1D0
      XK(2,4)=-.339981043584 9D0
      XK(3,4)=-XK(2,4)
      XK(4,4)=-XK(1,4)
      WGT(1,1)=2.0D0
      WGT(2,1)=0.0D0
      WGT(3,1)=0.0D0
      WGT(4,1)=0.0D0
      WGT(1,2)=1.0D0
      WGT(2,2)=1.0D0
      WGT(3,2)=0.0D0
      WGT(4,2)=0.0D0
      WGT(1,3)=.5555555555556D0
      WGT(2,3)=.8888888888889D0
      WGT(3,3)=.5555555555556D0
      WGT(4,3)=0.0D0
      WGT(1,4)=.3478548451375D0
      WGT(2,4)=.6521451548625D0
      WGT(3,4)=WGT(2,4)
      WGT(4,4)=WGT(1,4)
      IPERM(1)=2
      IPERM(2)=3
      IPERM(3)=1
C
C
C
C     ZERO EM
C
      WRITE (6,3000) NBRK8,NMAT,NLD
      DO 9 I=1,1058
    9 LM(I)=0
C
C     MATERIAL PROPERTIES
C
      WRITE (6,1300)
      DO 1 I=1,NMAT
      READ  (5,1001) N,EE(N),ENU(N),RHO(N),ALPT(N)
    1 WRITE (6,2001) N,EE(N),ENU(N),RHO(N),ALPT(N)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) (EE(I),ENU(I),RHO(I),ALPT(I),I=1,NMAT)
C
C     ELEMENT DISTRIBUTED LOAD CARDS
C
      IF(NLD) 23,23,15
   15 WRITE (6,1302)
      DO 16 I=1,NLD
      READ  (5,1002) N,KTYPE(N),PR(N),YREF(N),NFACE(N)
   16 WRITE (6,2002) N,KTYPE(N),PR(N),YREF(N),NFACE(N)
C***  DATA PORTHOLE SAVE
```

```
      IF(MODEX.EQ.1)
     *WRITE (NT8) (KTYPE(N),PR(N),YREF(N),NFACE(N),N=1,NLD)
C
   23 READ  (5,1003) GRAV,PLF,TLF,XLF,YLF,ZLF
      WRITE (6,2003) GRAV,PLF,TLF,XLF,YLF,ZLF
      IF(GRAV.EQ.0.) GRAV=1.E+10
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) GRAV,PLF,TLF,XLF,YLF,ZLF
C
      WRITE (6,1301)
      NEL=0
   30 READ  (5,1000) INEL,INP,ININT,IMAT,IINC,MLD,ISP,TTEMP
      IF(IINC.EQ.0) IINC=1
      IF(IMAT.EQ.0) IMAT=1
   40 NEL=NEL+1
      ML=INEL-NEL
      IF(ML) 50,55,60
   50 WRITE (6,4003) INEL
      STOP
   55 DO 56 I=1,8
   56 NP(I)=INP(I)
      DO 39 I=1,4
   39 MULT(I)=1
      NINT=ININT
      MAT=IMAT
      INC=IINC
      TAG=STAR
      REFT=TTEMP
      IS(1)=ISP(1)
      IS(2)=ISP(2)
      SKIP=99999.
      IF(NINT) 33,33,57
   33 NINT=IABS(NINT)
      SKIP=1.
      IF(NINT.EQ.0) SKIP=0.
   57 CONTINUE
      DO 59 I=1,4
      KLD(I)=IABS(MLD(I))
      IF(MLD(I)) 58,58,59
   58 MULT(I)=0
   59 CONTINUE
      GO TO 62
C
   60 DO 61 I=1,8
   61 NP(I)=NP(I)+INC
      TAG=BLNK
      DO 64 I=1,4
   64 KLD(I)=KLD(I)*MULT(I)
C
   62 IF(MODEX.EQ.1) GO TO 540
      TEMP = 0.0
      DO 10 I=1,8
      K=NP(I)
      TEMP=TEMP+T(K)
```

```
      XX(I,1)=X(K)
      XX(I,2)=Y(K)
   10 XX(I,3)=Z(K)
      TEMP=TEMP*0.125
      K=MAT
      FAC = EE(K)/((1.-2.*ENU(K))*(1.+ENU(K)))
      FACT=FAC*ALPT(K)*(TEMP-REFT)*(1.+ENU(K))
      IF(SKIP) 70,70,63
   63 SKIP=SKIP-1.
      CC1=1.-ENU(K)
      CC2=ENU(K)
      CC3=.5-ENU(K)
C
      DO 100 I=1,33
      DO 100 J=1,33
  100 S(I,J)=0.0D0
      DO 110 I=1,24
  110 TT(I)=0.
      DO 120 I=1,8
  120 DL(I)=0.
      VOLUME = 0.0
C
C     LOOP OVER NINT**3 INTEGRATION POINTS
C
      DO 300  LX = 1,NINT
      E1=XK(LX,NINT)
      DO 300  LY = 1,NINT
      E2=XK(LY,NINT)
      DO 300  LZ = 1,NINT
      E3=XK(LZ,NINT)
C
      CALL DERIV(1,SA)
C
      GT= WGT(LX,NINT)*WGT(LY,NINT)*WGT(LZ,NINT)*DET
      VOLUME = VOLUME + GT
      GG=GT*RHO(MAT)
      G=GT*FAC
      C1=G*CC1
      C2=G*CC2
      C3=G*CC3
C
      L=0
      DO 310 I=1,8
      DL(I)=DL(I) + GG*Q(I)
      DO 310 K=1,3
      L=L+1
  310 TT(L)=TT(L) + GT*SA(I,K)
C
C     ADD CONTRIBUTION TO STIFFNESS MATRIX
C
      DO 300 I=1,11
      K3 = 3*I
      K2 = K3 - 1
      K1 = K2 - 1
      UI=SA(I,1)
```

```
      VI=SA(I,2)
      WI=SA(I,3)
      DO 300 J=1,11
      L3 = 3*J
      L2 = L3 - 1
      L1 = L2 - 1
      UJ=SA(J,1)
      VJ=SA(J,2)
      WJ=SA(J,3)
      UU=UI*UJ
      VV=VI*VJ
      WW=WI*WJ
      UV=UI*VJ
      VU=VI*UJ
      UW=UI*WJ
      WU=WI*UJ
      VW=VI*WJ
      WV=WI*VJ
      S(K1,L1) = S(K1,L1) + C1*UU + C3*(VV+WW)
      S(K2,L2) = S(K2,L2) + C1*VV + C3*(WW+UU)
      S(K3,L3) = S(K3,L3) + C1*WW + C3*(UU+VV)
      S(K1,L2) = S(K1,L2) + C2*UV + C3*VU
      S(K1,L3) = S(K1,L3) + C2*UW + C3*WU
      S(K2,L3) = S(K2,L3) + C2*VW + C3*WV
      IF (I.EQ.J)  GO TO 300
      S(K2,L1) = S(K2,L1) + C2*VU + C3*UV
      S(K3,L1) = S(K3,L1) + C2*WU + C3*UW
      S(K3,L2) = S(K3,L2) + C2*WV + C3*VW
  300 CONTINUE
C
C     FORM STRAIN MATRIX
C
      NSS=2
      IF(IS(2).EQ.0) NSS=1
      DO 305 I=1,12
      DO 305 J=1,33
  305 STR(I,J)=0.
      DO 405 L=1,NSS
      LL=IS(L)+1
      E1=STPTS(LL,1)
      E2=STPTS(LL,2)
      E3=STPTS(LL,3)
      CALL DERIV(2,SA)
      L3=6*L-6
      DO 402 K=1,11
      K3=3*K
      K2=K3-1
      K1=K2-1
      STR(L3+1,K1) = SA(K,1)
      STR(L3+2,K2) = SA(K,2)
      STR(L3+3,K3) = SA(K,3)
      STR(L3+4,K1) = SA(K,2)
      STR(L3+4,K2) = SA(K,1)
      STR(L3+5,K2) = SA(K,3)
      STR(L3+5,K3) = SA(K,2)
```

```
      STR(L3+6,K1) = SA(K,3)
  402 STR(L3+6,K3) = SA(K,1)
  405 CONTINUE
      NS=6*NSS
C
C     STATIC CONDENSATION
C
      DO 710 M=1,9
      MN=34-M
      MO=MN-1
C      STIFFNESS MATRIX - S
      SP=S(MN,MN)
      DO 650 I=1,MO
  650 S(MN,I)=S(I,MN)/SP
      DO 700 K=1,MO
      SP=S(MN,K)
      DO 700 J=1,K
  700 S(J,K)=S(J,K) - SP*S(J,MN)
C      DERIVATIVE MATRIX - STR
      DO 710 J=1,NS
      SP=STR(J,MN)
      IF(SP.EQ.0.) GO TO 710
      DO 705 K=1,MO
  705 STR(J,K)=STR(J,K) - SP*S(MN,K)
  710 CONTINUE
C
      DO 760 I=1,24
      DO 760 J=1,24
      SS(I,J)=S(I,J)
  760 SS(J,I)=SS(I,J)
C
C     STRAIN TO STRESS MATRIX
C
      E(1,1)=CC1*FAC
      E(2,2)=E(1,1)
      E(3,3)=E(1,1)
      E(1,2)=CC2*FAC
      E(1,3)=E(1,2)
      E(2,3)=E(1,2)
      E(2,1)=E(1,2)
      E(3,1)=E(1,2)
      E(3,2)=E(1,2)
C
      DO 900 I=1,NSS
      II=I*6-6
      DO 850 J=1,3
      DO 850 K=1,24
      SP=0.
      DO 840 L=1,3
  840 SP = SP + E(J,L)*STR(II+L,K)
      SA(II+J,K)=SP
      JJ=II+3+J
  850 SA(JJ,K)=CC3*FAC*STR(JJ,K)
C
C
```

```
      DO 860 J=1,3
      JJ=J+3
      DO 860 K=1,4
      SF(II+J,K)=-FACT*TLF(K)
  860 SF(II+JJ,K)=0.
C
      IF (IS(I).LE.0) GO TO 900
      LL=IS(I)+1
      E1=STPTS(LL,1)
      E2=STPTS(LL,2)
      E3=STPTS(LL,3)
      CALL DERIV (4,SA)
      CALL LOSTR (IS,A,B,SA,SF,I)
C
  900 CONTINUE
C
 70   CONTINUE
C
C     DISTRIBUTED LOAD
C
      DO 410 J=1,24
      DO 410 I=1,4
  410 RF(J,I)=0.
      CALL LOAD (KTYPE,PR,YREF,NFACE)
C
C     SELF WQT.
C
      DO 460 II=1,8
      K=3*II
      J=K-1
      I=J-1
      DO 460 L=1,4
      RF(I,L) = RF(I,L)*PLF(L) + DL(II)*XLF(L)
      RF(J,L) = RF(J,L)*PLF(L) + DL(II)*YLF(L)
  460 RF(K,L) = RF(K,L)*PLF(L) + DL(II)*ZLF(L)
C
C     THERMAL LOADS
C
      DO 470 I=1,24
      GT=TT(I)*FACT
      DO 470 J=1,4
  470 RF(I,J)=RF(I,J) + GT*TLF(J)
C
C     MASS ARRAY
C
      L=0
      DUM=VOLUME*RHO(MAT)*.125/GRAV
      DO 465 I=1,8
      DO 465 J=1,3
      L=L+1
  465 XM(L) = DUM
C
  540 IJ = 0
      DO 550 I=1,8
      II=NP(I)
```

```
      DO 550 J=1,3
      IJ=IJ+1
  550 LM(IJ)=ID(II,J)
      ND=24
C
      IS1=IS(1)
      IS2=IS(2)
      NDM=24
      CALL CALBAN (MBAND,NDIF,LM,XM,SS,RF,ND,NDM,NS)
      IF(MODEX.EQ.1) GO TO 560
      WRITE (1) ND,NS,(LM(I),I=1,ND),((SA(I,J),I=1,NS),J=1,ND),
     1 ((SF(I,J),I=1,NS),J=1,4)
  560 IF(MODEX.EQ.1)
     *WRITE (NT8) NEL,NP,NINT,MAT,KLD,REFT,IS
      WRITE (6,2000) NEL,NP,NINT,MAT,TAG,KLD,REFT,IS,NDIF
C
C     CHECK IF LAST ELEMENT
C
      IF(NBRK8-NEL) 50,600,590
  590 IF(ML) 30,30,40
C
C
  600 RETURN
C
C
 1000 FORMAT (12I5,4I2,2I1,F10.2)
 1001 FORMAT (I5,4F10.0)
 1002 FORMAT (2I5,2F10.2,I5)
 1003 FORMAT(F10.2/(4F10.2))
 2000 FORMAT(I6,1X,8I5,I9,I12,8X,A1,3X,4I5,F9.1,5X,2I3,I8)
 2001 FORMAT(1X,I5,4E15.4)
 2002 FORMAT (I5,I9,2F13.3,I12)
 2003 FORMAT (/////
     . 35H .....ACCELERATION DUE TO GRAVITY =  F10.2////
     . 38H LOAD FACTORS FOR 4 ELEMENT LOAD CASES  //
     . 46X 17HELEMENT LOAD CASE /
     . 36X 1HA 9X 1HB 9X 1HC 9X 1HD  /
     . 30H PRESSURE LOAD FACTORS . .      4F10.3/
     . 30H THERMAL  LOAD FACTORS . .      4F10.3//
     . 30H PERCENT GRAVITY IN +X DIRN.   4F10.3/
     . 30H PERCENT GRAVITY IN +Y DIRN.   4F10.3/
     . 30H PERCENT GRAVITY IN +Z DIRN.   4F10.3/ )
 1300 FORMAT (9HOMATERIAL 10X 1HE 12X 2HNU 10X 3HRHO 11X 7HALPHA-T /
     . 8H  NUMBER /)
 1301 FORMAT (30H1....8 NODE SOLID ELEMENT DATA ///
     . 8H ELEMENT 10X 15HCONNECTED NODES 17X ,'28HINTEGRATION  MATERIALI
     .NPUT', 7X 13HELEMENT LOADS 5X 7HELEMENT ,5X,6HSTRESS /
     . 8H  NUMBER 3X,36H1     2     3     4     5     6     7     8 6X,5HORDER,
     . 7X,3HNO. 6X 3HTAG 7X 16H1     2     3     4 4X  5HTEMP. ,6X,6HPOINTS
     .,5X,4HBAND /)
 1302 FORMAT (/////26H ELEMENT DISTRIBUTED LOADS //
     . 52H NUMBER    KTYPE        PR            YREF           FACE )
 3000 FORMAT ( 31H1......8 - NODE SOLID ELEMENTS  ///
     . 24H  NUMBER OF ELEMENTS.... ,I5 //
     . 24H  NUMBER OF MATERIALS... ,I5 //
```

```
     . 24H  NUMBER OF LOAD TYPES.. ,I5 ///)
 4003 FORMAT (36HOELEMENT CARD ERROR, ELEMENT NUMBER= I6)
 4004 FORMAT ('ONUMBER OF DISPLACEMENTS PER ELEMENT (ND) =',I3,/,
     1        'ONUMBER OF STRESSES PER ELEMENT (NS)      =',I3,/,
     2        'OELEMENT STRESS-DISPL MATRIX?')
 4005 FORMAT (/,(1H ,1P10E13.4))
 4006 FORMAT ('OELEMENT FIXED-NODE STRESSES?',/,(1H ,1P4E13.4))
 4007 FORMAT ('ELEMENT',I3,'   ND=',I3,'   NS=',I3)
 4008 FORMAT ((1P8E10.3))
C
      END
      SUBROUTINE CLAMP (NUMEL,ID,X,Y,Z,NUMNP,MBAND)
C
C     CALLS?  CALBAN
C     CALLED BY?  BOUND
C
      COMMON/EM/LM(24),ND,NS,S(24,24),P(24,4),XM(24),SA(12,24),TT(12,4),
     1    IFILL1(3048)
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,1)
      COMMON / JUNK / R(6),RM(4),IFILL2(410)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      WRITE (6,2000) NUMEL
C
      NS=2
      ND=6
C
      READ  (5,1005) RM
      WRITE (6,2005) RM
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) RM
C
C     INITIALIZATION
C
      DO 30 NI=1,ND
      XM(NI)   = 0.0
      DO 20 NJ=1,ND
   20 S(NI,NJ)= 0.0
   30 CONTINUE
      DO 50 NK=1,NS
      DO 40 NL=1,ND
   40 SA(NK,NL)  = 0.0
      DO 50 NI=1,4
      TT(NK,NI)  = 0.0
   50 CONTINUE
C
      NE=0
      WRITE (6,2007)
  210 KG=0
      MARK=0
C
  200 READ (5,1000) NP,NI,NJ,NK,NL,KD,KR,KN,SD,SR,TRACE
```

```
      IF (TRACE.EQ.0.) TRACE=1.0E+10
      IF (KG.GT.0) GO TO 550
C
C     COMPUTE THE DIRECTION COSINES OF THE ELEMENT*S AXIS
C
      KG=KN
      IF(MODEX.EQ.1) GO TO 530
      IF(NJ.EQ.0) GO TO 250
      X1=X(NJ)-X(NI)
      Y1=Y(NJ)-Y(NI)
      Z1=Z(NJ)-Z(NI)
      X2=X(NL)-X(NK)
      Y2=Y(NL)-Y(NK)
      Z2=Z(NL)-Z(NK)
      T1=Y1*Z2-Y2*Z1
      T2=Z1*X2-Z2*X1
      T3=X1*Y2-X2*Y1
      GO TO 260
  250 T1=X(NI)-X(NP)
      T2=Y(NI)-Y(NP)
      T3=Z(NI)-Z(NP)
  260 XL=T1*T1+T2*T2+T3*T3
      XL=SQRT(XL)
      IF(XL.GT.1.0E-6) GO TO 270
      WRITE (6,3000)
 3000 FORMAT (32H0*** ERROR   ZERO ELEMENT LENGTH, / 1X)
      STOP
  270 CONTINUE
      T1=T1/XL
      T2=T2/XL
      T3=T3/XL
C
C     DISPLACEMENT PRESCRIPTION
C
      IF (KD.EQ.0) GO TO 300
      SA(1,1)=T1*TRACE
      SA(1,2)=T2*TRACE
      SA(1,3)=T3*TRACE
      S(1,1)=T1*T1*TRACE
      S(1,2)=T1*T2*TRACE
      S(1,3)=T1*T3*TRACE
      S(2,2)=T2*T2*TRACE
      S(2,3)=T2*T3*TRACE
      S(3,3)=T3*T3*TRACE
      PP=TRACE*SD
      R(1)=T1*PP
      R(2)=T2*PP
      R(3)=T3*PP
      GO TO 350
  300 DO 310 I=1,3
      R(I)     = 0.0
      SA(1,I)  = 0.0
      DO 310 J=1,3
  310 S(I,J)   = 0.0
C
```

```
C     ROTATION PRESCRIPTION
C
  350 IF  (KR.EQ.0) GO TO 400
      SA(2,5)=T2*TRACE
      SA(2,4)=T1*TRACE
      SA(2,6)=T3*TRACE
      S(4,4)=T1*T1*TRACE
      S(4,5)=T1*T2*TRACE
      S(4,6)=T1*T3*TRACE
      S(5,5)=T2*T2*TRACE
      S(5,6)=T2*T3*TRACE
      S(6,6)=T3*T3*TRACE
      PP=TRACE*SR
      R(4)=T1*PP
      R(5)=T2*PP
      R(6)=T3*PP
      GO TO 450
  400 DO 410 I=4,6
      R(I)     = 0.0
      SA(2,I)  = 0.0
      DO 410 J=1,6
  410 S(I,J)   = 0.0
  450 DO 500 I=1,ND
      DO 500 J=1,ND
  500 S(J,I)   = S(I,J)
      DO 520 I=1,ND
      DO 520 J=1,4
  520 P(I,J)=R(I)*RM(J)
  530 NN = NP
      NNI=NI
      NNJ=NJ
      NNK=NK
      NNL=NL
      NKD=KD
      NKR=KR
      SSD=SD
      SSR=SR
      TTR=TRACE
      GO TO 560
  550 MARK=1
  555 NN=NN+KG
      NNI=NNI+KG
  560 KEL = NE+1
      WRITE  (6,2010) KEL,NN,NNI,NNJ,NNK,NNL,NKD,NKR,KN,SSD,SSR,TTR
      NE=NE+1
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE  (NT8) NE,NN,NNI,NNJ,NNK,NNL,NKD,NKR,SSD,SSR,TTR
C
      DO 600 I=1,ND
  600 LM(I)=ID(NN,I)
C
      NDM=24
      CALL CALBAN (MBAND,NDIF,LM,XM,S,P,ND,NDM,NS)
      IF(MODEX.EQ.1) GO TO 650
```

```
      WRITE (1) ND,NS,(LM(L),L=1,ND),((SA(L,K),L=1,NS),K=1,ND),
     1 ((TT(L,K),L=1,NS),K=1,4)
C
  650 CONTINUE
      IF (NE.EQ.NUMEL) RETURN
      IF (NN.LT.NP) GO TO 555
      IF (MARK.EQ.1) GO TO 210
      GO TO 200
C
 1000 FORMAT (8I5,3F10.0)
 1005 FORMAT (4F10.0)
C
 2000 FORMAT (34H1B O U N D A R Y   E L E M E N T S, ///
     1        27H ELEMENT TYPE        =      7, /
     2        21H NUMBER OF ELEMENTS =,I6      /// 1X)
 2005 FORMAT (30H ELEMENT LOAD CASE MULTIPLIERS, // 8X,7HCASE(A),8X,
     1        7HCASE(B),8X,7HCASE(C),8X,7HCASE(D),/ 4F15.4 /// 1X)
 2007 FORMAT (53H ELEMENT   NODE   NODES DEFINING CONSTRAINT DIRECTION,
     1     3X,38HCODE   CODE  GENERATION      SPECIFIED,6X,
     2        22HSPECIFIED      SPRING, /
     3        53H  NUMBER     (N)        (NI)      (NJ)      (NK)      (NL),
     4     3X,38H  KD      KR   CODE (KN)   DISPLACEMENT,6X,
     5        22H ROTATION          RATE, / 1X)
 2010 FORMAT (1X,2(2X,I5),2X,4(4X,I5),2(2X,I5),7X,I5,2E15.4,E13.4)
      END
      SUBROUTINE CROSS(A,B,C)
C
C     CALLED BY?  PLNAX
C
      DIMENSION A(4),B(4),C(4)
      X=A(2)*B(3)-A(3)*B(2)
      Y=A(3)*B(1)-A(1)*B(3)
      Z=A(1)*B(2)-A(2)*B(1)
      C(4)=SQRT(X*X+Y*Y+Z*Z)
      C(3)=Z/C(4)
      C(2)=Y/C(4)
      C(1)=X/C(4)
      RETURN
      END
      SUBROUTINE CROSS2 (A,B,C,IERR)
C
C     CALLED BY ? INP21
C
C
C     THIS ROUTINE FORMS THE VECTOR PRODUCT   C = A*B   WHERE *C*
C     IS NORMALIZED TO UNIT LENGTH
C
      DIMENSION A(3),B(3),C(3)
C
      X = A(2) * B(3) - A(3) * B(2)
      Y = A(3) * B(1) - A(1) * B(3)
      Z = A(1) * B(2) - A(2) * B(1)
      XLN =SQRT(X*X+Y*Y+Z*Z)
      IERR = 1
      IF(XLN.LE.1.0E-08) RETURN
```

```
      XLN = 1.0 /XLN
      C(3)  = Z * XLN
      C(2)  = Y * XLN
      C(1)  = X * XLN
      IERR = 0
      RETURN
      END
      SUBROUTINE CSTSTR (SCST,XST)
C
C     CALLED BY?  STRETR
C
C     THIS SUBROUTINE FORMS THE STRESS/DISPLACEMENT TRANSFORMATION
C     MATRIX FOR A CONSTANT STRAIN TRIANGLE (CST)
C
C**** I N P U T S
C
C     A,B,C          AS IN SLST.
C
C**** O U T P U T S
C
C     SCST(I,J)      I=1...3,J=1...6.  MEMBRANE STRESSES  SIG(XX)/(I=1),
C                    SIG(YY)/(I=2), SIG(XY)/(I=3).  IN-PLANE NODAL
C                    DISPLACEMENTS  U(1)/(J=1), U(2)/(J=2), U(3)/(J=3),
C                    V(1)/(J=4), V(2)/(J=5), V(3)/(J=6).
C
      COMMON /TRIARG/
     1 A(3),B(3),H(3),HPT(3),C(3,3),SMT(3,3),BMT(3,3),
     2 U(6),V(6),W(3),RX(3),RY(3),RM(3),ST(12,12)
C
      DIMENSION    SCST(3,6),XST(3,6)
C
      DO 10 I=1,3
      XST(1,I+3) = 0.0
   10 XST(2,I  ) = 0.0
C
      AREA = A(3)* B(2) - A(2)* B(3)
      IF(AREA.LT.1.0E-8) STOP 100
      DUM = 1.0/AREA
C     STRAIN-DISPLACEMENT
      DO 20 K=1,3
      XST(1,K  ) = B(K)* DUM
      XST(2,K+3) = A(K)* DUM
      XST(3,K  ) = A(K)* DUM
   20 XST(3,K+3) = B(K)* DUM
C     STRESS-DISPLACEMENT
      DO 50 I=1,3
      DO 40 J=1,6
      SCST(I,J) = 0.0
      DO 30 K=1,3
   30 SCST(I,J) = SCST(I,J) + C(I,K)* XST(K,J)
   40 CONTINUE
   50 CONTINUE
C
      RETURN
      END
```

```
      SUBROUTINE DECOMP (A,B,MAXA,NEQB,MA,NBLOCK,NWA,NTB,NSCH,NEQ,MI)
C
C     CALLED BY?  SSPCEB
C
      COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
      DIMENSION A(NWA),B(NWA),MAXA(MI)
C
      MA2=MA - 2
      IF(MA2.EQ.0) MA2=1
      INC=NEQB - 1
      N1=NL
      N2=NT
      REWIND NSTIF
      REWIND NRED
      REWIND N1
      REWIND N2
      NSCH=0
C
C     MAIN LOOP OVER ALL BLOCKS
      DO 600 NJ=1,NBLOCK
      IF (NJ.NE.1) GO TO 10
      READ (NSTIF) A
      GO TO 100
 10   IF (NTB.EQ.1) GO TO 100
      REWIND N1
      REWIND N2
      READ (N1) A
C
C     FIND COLUMN HEIGHTS
 100  KU=1
      KM=MINO(MA,NEQB)
      MAXA(1)=1
      DO 110 N=2,MI
      IF (N-MA) 120,120,130
 120  KU=KU + NEQB
      KK=KU
      MM=MINO(N,KM)
      GO TO 140
 130  KU=KU + 1
      KK=KU
      IF (N-NEQB) 140,140,136
 136  MM=MM - 1
 140  DO 160 K=1,MM
      IF (A(KK)) 110,160,110
 160  KK=KK - INC
 110  MAXA(N)=KK
C
      IF (A(1)) 172,174,176
 174  KK=(NJ-1)*NEQB + 1
      IF (KK.GT.NEQ) GO TO 590
      WRITE (6,1000) KK
      STOP
 172  NSCH=NSCH + 1
C
C     FACTORIZE LEADING BLOCK
```

```
 176    DO 200 N=2,NEQB
        NH=MAXA(N)
       IF (NH - N) 200,200,210
 210    KL=N + INC
        KU=NH
        K=N
        D=0.
        DO 220 KK=KL,KU,INC
        K=K - 1
        C=A(KK)/A(K)
        D=D + C*A(KK)
 220    A(KK)=C
        A(N)=A(N) - D
C
  245  IF (A(N)) 222,224,230
 224    KK=(NJ-1)*NEQB + N
        IF (KK.GT.NEQ) GO TO 590
       WRITE (6,1000) KK
        STOP
 222    NSCH=NSCH + 1
C
 230    IC=NEQB
        DO 240 J=1,MA2
        MJ=MAXA(N+J) - IC
        IF (MJ-N) 240,240,280
 280    KU=MINO(MJ,NH)
        KN=N + IC
        C=0.
        DO 300 KK=KL,KU,INC
 300    C=C + A(KK)*A(KK+IC)
        A(KN)=A(KN) - C
 240    IC=IC + NEQB
C
 200    CONTINUE
C
C       CARRY OVER INTO TRAILING BLOCKS
 320    DO 400 NK=1,NTB
        IF ((NK+NJ).GT.NBLOCK) GO TO 400
        NI=N1
        IF ((NJ.EQ.1).OR.(NK.EQ.NTB)) NI=NSTIF
        READ (NI) B
        ML=NK*NEQB + 1
        MR=MINO((NK+1)*NEQB,MI)
        MD=MI - ML
        KL=NEQB + (NK-1)*NEQB*NEQB
        N=1
C
        DO 500 M=ML,MR
        NH=MAXA(M)
        KL=KL + NEQB
        IF (NH-KL) 505,510,510
 510    KU=NH
        K=NEQB
        D=0.
        DO 520 KK=KL,KU,INC
```

```
      C=A(KK)/A(K)
      D=D + C*A(KK)
      A(KK)=C
  520 K=K - 1
      B(N)=B(N) - D
      IF (MD) 500,500,530
  530 IC=NEQB
      DO 540 J=1,MD
      MJ=MAXA(M+J) - IC
      IF (MJ-KL) 540,550,550
  550 KU=MINO(MJ,NH)
      KN=N + IC
      C=0.
      DO 575 KK=KL,KU,INC
  575 C=C + A(KK)*A(KK+IC)
      B(KN)=B(KN) - C
  540 IC=IC + NEQB
  505 MD=MD - 1
C
  500 N=N + 1
C
      IF (NTB.NE.1) GO TO 560
      WRITE (NRED) A,MAXA
      DO 570 I=1,NWA
  570 A(I)=B(I)
      GO TO 600
  560 WRITE (N2) B
C
  400 CONTINUE
C
      M=N1
      N1=N2
      N2=M
  590 WRITE (NRED) A,MAXA
C
  600 CONTINUE
C
 1000 FORMAT (37H0***ERROR   SOLUTION STOP IN *DECOMP*, / 12X,
     1          37HZERO PIVOT FOUND DURING FACTORIZATION, / 12X,
     2          17HEQUATION NUMBER =, I5 / 1X)
C
      RETURN
      END
      SUBROUTINE DERIV(KK,D)
C
C     CALLED BY?  BRICK8,LOAD
C
      DIMENSION D(12,1)
      COMMON /GASS/ XK(4,4),WGT(4,4),IPERM(3)
      COMMON /JUNK/ R ,S ,T ,DET,MLD(4),KLD(4),MULT(4),NP(8),INP(8),
     .              A(3,3),P(3,11),B(3,3),XX(8,3),Q(11),DL(8),IFILL(206)
C
      RP=(1.+R)*.125
      RM=(1.-R)*.125
      SP=1.+S
```

```
      SM=1.-S
      TP=1.+T
      TM=1.-T
      IF (KK.EQ.2.OR.KK.EQ.4) GO TO 100
C
C     SHAPE FUNCTIONS
C
      Q(1)  = RP*SM*TM
      Q(2)  = RP*SP*TM
      Q(3)  = RM*SP*TM
      Q(4)  = RM*SM*TM
      Q(5)  = RP*SM*TP
      Q(6)  = RP*SP*TP
      Q(7)  = RM*SP*TP
      Q(8)  = RM*SM*TP
C
C     DERIVATIVES OF SHAPE FUNCTIONS
C

  100 P(1,1)  = SM*TM*.125
      P(1,2)  = SP*TM*.125
      P(1,3)  = -P(1,2)
      P(1,4)  = -P(1,1)
      P(1,5)  = SM*TP*.125
      P(1,6)  = SP*TP*.125
      P(1,7)  = -P(1,6)
      P(1,8)  = -P(1,5)
      P(1,9)  = -R
      P(1,10) = 0.
      P(1,11) = 0.
C
      P(2,1)  = -RP*TM
      P(2,2)  = -P(2,1)
      P(2,3)  = RM*TM
      P(2,4)  = -P(2,3)
      P(2,5)  = -RP*TP
      P(2,6)  = -P(2,5)
      P(2,7)  = RM*TP
      P(2,8)  = -P(2,7)
      P(2,9)  = 0.
      P(2,10) = -S
      P(2,11) = 0.
C
      P(3,1)  = -RP*SM
      P(3,2)  = -RP*SP
      P(3,3)  = -RM*SP
      P(3,4)  = -RM*SM
      P(3,5)  = -P(3,1)
      P(3,6)  = -P(3,2)
      P(3,7)  = -P(3,3)
      P(3,8)  = -P(3,4)
      P(3,9)  = 0.
      P(3,10) = 0.
      P(3,11) = -T
C
```

```
C      JACOBIAN MATRIX A
C
       DO 200 I=1,3
       DO 200 J=1,3
       C=0.
       DO 150 L=1,8
  150  C = C + P(I,L)*XX(L,J)
  200  A(I,J) = C
C
C      INVERT JACOBIAN
C
       IF(KK.EQ.3) GO TO 500
       DO 250 I=1,3
       J = IPERM(I)
       K = IPERM(J)
       B(I,I) = A(J,J)*A(K,K) - A(K,J)*A(J,K)
       B(I,J) = A(K,J)*A(I,K) - A(I,J)*A(K,K)
  250  B(J,I) = A(J,K)*A(K,I) - A(J,I)*A(K,K)
       IF (KK.EQ.4) GO TO 500
       DET = A(1,1)*B(1,1) + A(1,2)*B(2,1) + A(1,3)*B(3,1)
C
C      MATRIX OF X-Y-Z DERIVATIVES
       DO 400 I=1,3
       DO 400 J=1,11
       C = 0.
       DO 350 K=1,3
  350  C = C + B(I,K)*P(K,J)
  400  D(J,I)=C/DET
C
  500  RETURN
C
       END
       SUBROUTINE DER3DS (NEL,XX,B,DET,R,S,T,NOD9,H,P,IELD,IELX)
C
C      CALLED BY ? THDFE
C
C
C
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                                   .
C .    P R O G R A M                                                  .
C .                                                                   .
C .       EVALUATES STRAIN-DISPLACEMENT MATRIX B AT POINT (R,S,T)     .
C .                                                                   .
C .       CURVILINEAR HEXAHEDRON    8 TO 21 NODES                     .
C .                                                                   .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C
C
       DIMENSION  XX(3,1),B(6,1),NOD9(1),H(1),P(3,1)
       DIMENSION  XJ(3,3),XJI(3,3)
C
C
```

```
C      FIND INTERPOLATION FUNCTIONS AND THEIR DERIVATIVES
C      EVALUATE JACOBIAN MATRIX AT POINT (R,S,T)
C      COMPUTE DETERMINANT OF JACOBIAN MATRIX AT POINT (R,S,T)
C
C
       CALL FNCT (R,S,T,H,P,NOD9,XJ,DET,XX,IELD,IELX,NEL)
C
C
C      COMPUTE INVERSE OF JACOBIAN MATRIX
C
C
       DUM=1.0/DET
       XJI(1,1)=DUM*( XJ(2,2)*XJ(3,3)  - XJ(2,3)*XJ(3,2))
       XJI(2,1)=DUM*(-XJ(2,1)*XJ(3,3)  + XJ(2,3)*XJ(3,1))
       XJI(3,1)=DUM*( XJ(2,1)*XJ(3,2)  - XJ(2,2)*XJ(3,1))
       XJI(1,2)=DUM*(-XJ(1,2)*XJ(3,3)  + XJ(1,3)*XJ(3,2))
       XJI(2,2)=DUM*( XJ(1,1)*XJ(3,3)  - XJ(1,3)*XJ(3,1))
       XJI(3,2)=DUM*(-XJ(1,1)*XJ(3,2)  + XJ(1,2)*XJ(3,1))
       XJI(1,3)=DUM*( XJ(1,2)*XJ(2,3)   - XJ(1,3)*XJ(2,2))
       XJI(2,3)=DUM*(-XJ(1,1)*XJ(2,3)  + XJ(1,3)*XJ(2,1))
       XJI(3,3)=DUM*( XJ(1,1)*XJ(2,2)  - XJ(1,2)*XJ(2,1))
C
C
C      EVALUATE B MATRIX IN GLOBAL (X,Y,Z) COORDINATES
C
C
       DO 130 K=1,IELD
       K2=K*3
       DO 115 L=1,3
       B(L,K2-2) = 0.0
       B(L,K2-1) = 0.0
   115 B(L,K2  ) = 0.0
C
C      DIRECT STRAINS  (1=EXX, 2=EYY, 3=EZZ)
C
       DO 120 I=1,3
       B(1,K2-2) = B(1,K2-2) + XJI(1,I)* P(I,K)
       B(2,K2-1) = B(2,K2-1) + XJI(2,I)* P(I,K)
   120 B(3,K2  ) = B(3,K2  ) + XJI(3,I)* P(I,K)
C
C      SHEAR STRAINS  (4=EXY, 5=EYZ, 6=EZX)
C
       B(4,K2-2) = B(2,K2-1)
       B(4,K2-1) = B(1,K2-2)
       B(5,K2-1) = B(3,K2  )
       B(5,K2  ) = B(2,K2-1)
       B(6,K2-2) = B(3,K2  )
   130 B(6,K2  ) = B(1,K2-2)
C
C
       RETURN
C
       END
       SUBROUTINE DISPLR  (ID,F,FI,X,NEQB,NF,NDS,NUMNP,NBLOCK,NSB)
C
```

```
C     CALLS?  DISPLY
C     CALLED BY?  HISTRY
C
      DIMENSION ID(NUMNP,6),F(8,NF),FI(NSB ,NF),X(NF,NDS)
      COMMON /JUNK/ D(8),DDT,TIME,DD,XUM,DM(8),TM(8),NP,IC(6),L,II,
     1         MSB,NS,NE,N,M,J,K,MM,KD(3,8),IEQ,NRD,IFILL1(331)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
      COMMON / DYN / NT,NOT,DAMP,DT,IFILL3(6)
C
C     EQUATION NUMBERS OF SELECTED DISPLACEMENT COMPONENTS
C
      IF(MODEX.EQ.1) GO TO 50
      REWIND 9
      REWIND 8
      READ (8) ID
   50 L=0
      NUM=0
      READ (5,2000) KKK,ISP
      WRITE (6,1005)
  100 READ (5,2000) NP,IC
      WRITE (6,2001) NP,IC
      IF(MODEX.EQ.1 .AND. NP.EQ.0) GO TO 210
      IF(MODEX.EQ.1) GO TO 100
      IF(NP.GT.0) GO TO 110
      IF(L.EQ.0) GO TO 200
      WRITE (9) KD,L
      NUM=NUM+1
      GO TO 200
  110 DO 150 I=1,6
      II=IC(I)
      IF(II.EQ.0) GO TO 100
  120 L=L+1
      KD(1,L)=NP
      KD(2,L)=II
      KD(3,L)=ID(NP,II)
      IF(ID(NP,II).LE.0) L=L-1
      IF(L.LT.8) GO TO 150
      WRITE (9) KD,L
      NUM=NUM+1
      L=0
  150 CONTINUE
      GO TO 100
C
C     APPROPRIATE MODE SHAPE COMPONENTS
C
  200 IF(NUM.EQ.0) RETURN
  210 WRITE (6,4000) KKK,ISP
      IF(MODEX.EQ.1) RETURN
      REWIND 3
      REWIND 9
      REWIND 7
      READ (7)
      NE=NSB
      NS=NE+1-NEQB
      DO 300 I=1,NBLOCK
```

```
      READ (7) ((FI(J,K),J=NS,NE),K=1,NF)
      NS=NS-NEQB
  300 NE=NE-NEQB
C
      DO 400 N=1,NUM
      READ (9) KD,L
C
      DO 350 I=1,L
      II=KD(3,I)
      DO 350 J=1,NF
  350 F(I,J)=FI(II,J)
  400 WRITE (3) L,KD,F
C
C     COMPUTE AND OUTPUT HISTORY OF VALUES
C
  410 DT=NOT*DT
C
      CALL DISPLY (X,F,NF,NDS,NUM,1,KKK,2,ISP)
C
  900 RETURN
C
 1005 FORMAT (23H1DISPLACEMENT COMPONENT,/ 22H TIME HISTORY REQUESTS,//
     1 3X,4HNODE,3X,24HNODAL DEGREES OF FREEDOM,/ 7H NUMBER,3X,6(3X,
     2 1H*), / 1X)
 2000 FORMAT (7I5)
 2001 FORMAT (I7,3X,6I4)
 4000 FORMAT (// 25H CODE FOR OUTPUT TYPE   =, I2 /
     1           3X,19HEQ.1, HISTORY TABLE,       /
     2           3X,18HEQ.2, PRINTER PLOT,        /
     3           3X,17HEQ.3, MAXIMA ONLY,         /
     4           25H PRINTER PLOT SPACING   =, I2 / 1X)
C
      END
      SUBROUTINE DISPLY (X,F,NF,NDS,NUM,NN,KKK,ISD,ISP)
C
C     CALLS?  ELOUTH,PPLOT
C     CALLED BY?  DISPLR,STRSD1
C
C     SUBROUTINE TO PRINT RESPONSE TABLES, TO PRODUCE PRINTER PLOTS
C     OF DISPLACEMENT OR STRESS COMPONENTS, OR TO RECOVER MAXIMA ONLY
C
C     ISD = 1, STRESSES          KKK = 1, PRINT RESPONSE TABLES + MAXIMA
C     ISD = 2, DISPLACEMENTS     KKK = 2, PRINTER PLOTS         + MAXIMA
C                                KKK = 3, RECOVER                 MAXIMA
C
      DIMENSION X(NF,NDS),F(8,NF),NUM(NN)
      COMMON / JUNK / KD(3,8),TM(8),DM(8),D(8),IFILL1(358)
      COMMON / DYN / NT,NOT,DAMP,DT,IFILL2(6)
      COMMON / ELPAR / NPAR(14),IFILL3(10)
C
      REWIND 3
      REWIND 4
      READ (4) X
C
      DO 900 N=1,NN
```

```
      REWIND 2
      REWIND 9
      MM=NUM(N)
C
      IF(ISD.EQ.2) GO TO 90
      READ (3) NPAR
      MTYPE=NPAR(1)
   90 IF(MM.EQ.0) GO TO 900
C
      DO 600 M=1,MM
      IF (ISD.EQ.2) GO TO 70
      READ (3) L,KD,F,NS
      GO TO 80
   70 READ (3) L,KD,F
   80 GO TO (100,300,200),KKK
C
C     PRINT
C
  100 IF(ISD.EQ.1) GO TO 130
      WRITE (6,1000) M
      GO TO 140
  130 CALL ELOUTH (KD,L,MTYPE,M,NS)
      GO TO 300
  140 WRITE (6,2001) (KD(1,I),KD(2,I),I=1,L)
      GO TO 300
C
C     MAXIMUMS
C
  200 IF(M.GT.1) GO TO 300
      IF(ISD.EQ.1) GO TO 230
      WRITE (6,1002)
      WRITE (6,5001)
      GO TO 300
  230 WRITE (6,2002) MTYPE
      WRITE (6,4001)
C
C     COMPUTE HISTORY
C
  300 DO 320 I=1,L
      TM(I)=0.
  320 DM(I)=0.
      TIME=0.
C
      DO 500 K=1,NDS
      TIME=TIME + DT
      DO 450 I=1,L
      DD=0.
      DO 440 J=1,NF
  440 DD = DD + F(I,J)*X(J,K)
C
      AD=ABS(DD)
      IF(AD-DM(I)) 450,450,445
  445 DM(I)=AD
      TM(I)=TIME
C
```

```
  450 D(I)=DD
      GO TO (480,490,500),KKK
C
  480 WRITE (6,1004) TIME,(D(I),I=1,L)
      GO TO 500
C
  490 WRITE (9) D
C
  500 CONTINUE
C
      GO TO (510,520,530),KKK
C
  510 WRITE (6,1005) (DM(I),I=1,L)
      WRITE (6,1006) (TM(I),I=1,L)
      GO TO 600
C
  520 WRITE (2) KD,DM,TM,L
      GO TO 600
C
  530 WRITE (6,1007) (KD(1,I),KD(2,I),DM(I),TM(I),I=1,L)
C
  600 CONTINUE
C
C     PLOT SET OF VALUES
C
      IF(KKK.NE.2) GO TO 900
      REWIND 2
      REWIND 9
      DO 800 M=1,MM
      GO TO (610,620),ISD
C
  610 WRITE (6,4000) MTYPE,M
      WRITE (6,4001)
      GO TO 630
C
  620 WRITE (6,5000) M
      WRITE (6,5001)
C
  630 CALL PPLOT(2,9,NDS,ISP)
C
  800 CONTINUE
  900 CONTINUE
C
C
      RETURN
C
 1000 FORMAT (50H1D I S P L A C E M E N T   T I M E   H I S T O R Y, //
     1 13H OUTPUT SET =,I4, // 14X,27H*NODE NUMBER* - (COMPONENT ,
     2  7HNUMBER), 1X)
 1002 FORMAT (38H1D I S P L A C E M E N T  M A X I M A, // 1X)
 1004 FORMAT (F12.5,2X,1P8E12.3)
 1005 FORMAT (/ 24H MAXIMUM ABSOLUTE VALUES, // 8H MAXIMUM,6X,1P8E12.3)
 1006 FORMAT (5H TIME,9X,1P8E12.3)
 1007 FORMAT (I8,12X,I3,1P2E14.4,7X,2HNA)
 2001 FORMAT (8X,4HTIME,2X, 8(4X,I4,2H-(,I1,1H)) / 1X)
```

```
 2002 FORMAT (46H1S T R E S S   C O M P O N E N T   M A X I M A, //
     1         22H ELEMENT TYPE NUMBER =, I3, // 1X)
 4000 FORMAT (51H1N O R M A L I Z E D   S T R E S S   H I S T O R Y,3X,
     1  7HP L O T, // 22H ELEMENT TYPE NUMBER =, I3 /
     2                22H OUTPUT SET NUMBER   =, I3 // 1X)
 4001 FORMAT (8H ELEMENT,9X,6HSTRESS,7X,7HMAXIMUM,7X,7HTIME AT,5X,
     1 4HPLOT,/ 8H  NUMBER,6X,9HCOMPONENT,9X,5HVALUE,7X,7HMAXIMUM,3X,
     2 6HSYMBOL, / 1X)
 5000 FORMAT (46H1N O R M A L I Z E D   D I S P L A C E M E N T,3X,
     1 23HH I S T O R Y   P L O T, // 22H OUTPUT SET NUMBER   =, I3//1X)
 5001 FORMAT (4X,4HNODE,3X,12HDISPLACEMENT,7X,7HMAXIMUM,7X,7HTIME AT,
     2 5X,4HPLOT, / 8H  NUMBER,6X,9HCOMPONENT,9X,5HVALUE,7X,7HMAXIMUM,
     3 3X,6HSYMBOL, / 1X)
C
      END
      FUNCTION DOT(A,B)
C
C     CALLED BY?  PLNAX
C
      DIMENSION A(4),B(4)
      DOT=A(1)*B(1)+A(2)*B(2)+A(3)*B(3)
      RETURN
      END
       SUBROUTINE EIGSOL (DL,RTOLV,AR,BR,VEC,VL,VR,D,XM,NF,NV,NBLOCK,
     1NEQB,NITE,IFPR,NITEM,RTOL,IFSS,COFQ)
      REAL T1,T2
C
C     CALLS?  JACOBI
C     CALLED BY?  SSPCEB
C
       COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
       DIMENSION AR(NV,NV),BR(NV,NV),VEC(NV,NV),VL(NEQB,NV),VR(NEQB,NV)
       DIMENSION D(NV),DL(NV),RTOLV(NV),XM(NEQB)
C
       TOLJ=1.0E-12
       REWIND NMASS
       REWIND NT
       REWIND NR
C
C   FIND PROJECTIONS OF MASS AND STIFFNESS OPERATORS
       DO 100 I=1,NV
       DO 100 J=1,NV
       AR(I,J)=0.0
  100  BR(I,J)=0.0
       DO 200 N=1,NBLOCK
       BACKSPACE NL
       READ (NL) VL
       BACKSPACE NL
       READ (NR) VR
       READ (NMASS) XM
       DO 220 I=1,NV
       DO 220 J=1,NV
       ART=0.0
       DO 230 K=1,NEQB
  230  ART=ART+VL(K,I)*VR(K,J)
```

```
 220    AR(I,J)=AR(I,J)+ART
        DO 240 I=1,NEQB
        XMM=XM(I)
        DO 240 J=1,NV
 240    VR(I,J)=VL(I,J)*XMM
        WRITE (NT) VR
        DO 260 I=1,NV
        DO 260 J=1,NV
        BRT=0.0
        DO 280 K=1,NEQB
 280    BRT=BRT+VL(K,I)*VR(K,J)
 260    BR(I,J)=BR(I,J)+BRT
 200    CONTINUE
        DO 290 I=1,NV
        DO 290 J=1,I
        AR(I,J)=AR(J,I)
 290    BR(I,J)=BR(J,I)
C
C   SOLVE EIGENVALUE PROBLEM
        IF (IFPR.EQ.0) GO TO 293
        WRITE (6,1010)
        DO 292 I=1,NV
 292    WRITE (6,1000) (AR(I,J),J=1,NV)
        WRITE (6,1020)
        DO 294 I=1,NV
 294    WRITE (6,1000) (BR(I,J),J=1,NV)
C***       CALL TTIME (T1)
C
  293 CALL JACOBI (AR,BR,VEC,D,VL,NV,TOLJ,IFPR)
        DO 295 J=1,NV
        IF (BR(J,J).GT.0.) GO TO 291
        WRITE (6,1070)
      WRITE (6,1010)
      DO 501 L1=1,NV
  501 WRITE (6,1000) (AR(L1,L2),L2=1,NV)
      WRITE (6,1020)
      DO 502 L1=1,NV
  502 WRITE (6,1000) (BR(L1,L2),L2=1,NV)
        STOP
  291 XMM=SQRT(BR(J,J))
        DO 295 K=1,NV
 295    VEC(K,J)=VEC(K,J)/XMM
C
        IF (IFPR.EQ.0) GO TO 310
C***       CALL TTIME (T2)
        T3=T2 - T1
        WRITE (6,1080) T3
        WRITE (6,1030)
        WRITE (6,1010)
        DO 296 I=1,NV
 296    WRITE (6,1000) (AR(I,J),J=1,NV)
        WRITE (6,1020)
        DO 298 I=1,NV
 298    WRITE (6,1000) (BR(I,J),J=1,NV)
C
```

```
C   ARRANGE EIGENVALUES
 310    NV1=NV-1
 440    IS=0
        DO 400 I=1,NV1
        IF  (D(I+1).GE.D(I))  GO TO 400
        IS=IS+1
        BT=BR(I+1,I+1)
        DT=D(I+1)
        BR(I+1,I+1)=BR(I,I)
        D(I+1)=D(I)
        BR(I,I)=BT
        D(I)=DT
        DO 420 K=1,NV
        TEMP=VEC(K,I+1)
        VEC(K,I+1)=VEC(K,I)
 420    VEC(K,I)=TEMP
 400    CONTINUE
        IF  (IS.GT.0)  GO TO 440
C
C   CHECK FOR CONVERGENCE
        DO 300 I=1,NV
        IF  (D(I).GT.0.)  GO TO 302
        WRITE  (6,1090)
        STOP
  302 DIF=ABS(DL(I)-D(I))
 300    RTOLV(I)=DIF/D(I)
      IF(IFPR.EQ.0)  GO TO 304
      WRITE  (6,1040)
      WRITE  (6,1000)  (RTOLV(I),I=1,NV)
  304 CONTINUE
        DO 305 L=1,NF
        IF  (D(L).LT.COFQ)  GO TO 305
        IF  (RTOLV(L).GT.RTOL)  GO TO 306
        NF=L
        GO TO 306
 305    CONTINUE
 306    DO 320 I=1,NF
        IF  (RTOLV(I).GT.RTOL)  GO TO 340
 320    CONTINUE
      WRITE  (6,1050)  NF,RTOL
        NITE=NITEM
        GO TO 350
  340 IF(NITE.EQ.NITEM-2)  IFPR=1
      IF(NITE.LT.NITEM  )  GO TO 360
      WRITE  (6,1060)
        IFSS=1
 350    DO 354  I=1,NV
        DL(I)=D(I)
  354 D(I)=SQRT(D(I))
        M=NT
        NT=NL
        NL=M
        M=NR
        NR=NL
        NL=M
```

```
      REWIND NR
      WRITE (NR) (D(I),I=1,NF)
      GO TO 430
C
C   CALCULATE APPROXIMATE EIGEN DIRECTIONS
 360  DO 410 I=1,NV
 410  DL(I)=D(I)
      REWIND NR
 430  REWIND NT
      DO 460 N=1,NBLOCK
      READ (NT) VR
      DO 480 J=1,NV
      DO 480 I=1,NEQB
      TEMP=0.0
      DO 500 K=1,NV
 500  TEMP=TEMP+VR(I,K)*VEC(K,J)
 480  VL(I,J)=TEMP
 460  WRITE (NR) VL
C
      RETURN
 1000 FORMAT (1H ,12E11.4)
 1002 FORMAT (1H0,6E20.14)
 1010 FORMAT (10H0MATRIX AR )
 1020 FORMAT (10H0MATRIX BR )
 1030 FORMAT (40H0AR AND BR AFTER JACOBI DIAGONALIZATION  )
 1040 FORMAT (52H0RELATIVE TOLERANCE REACHED ON EIGENVALUES IS NOW    )
 1050 FORMAT (33H0CONVERGENCE ACHIEVED IN *EIGSOL*, /
     1        27H   NUMBER OF EIGENVALUES = , I3 /
     2        27H   RELATIVE TOLERANCE    = , E12.4 // 1X)
 1060 FORMAT (52H0WE ACCEPT THE CURRENT EIGENVALUE APPROXIMATIONS     )
 1070 FORMAT (37H0***ERROR   SOLUTION STOP IN *EIGSOL*, / 12X,
     1        39HNEGATIVE DIAGONAL ELEMENT IN MATRIX BR., // 1X)
 1080 FORMAT (28H0TIME FOR JACOBI ITERATION   F10.4)
 1090 FORMAT (37H0***ERROR   SOLUTION STOP IN *EIGSOL*, / 12X,
     1        44HINADMISSIBLE NEGATIVE EIGENVALUE CALCULATED., / 1X)
C
      END
      SUBROUTINE ELAW (NUMTC,EE,E,C,P,ALP)
C
C     CALLS?  POSINV
C     CALLED BY?  PLNAX
C
      COMMON /JUNK/ MAT,NT,TEMP,REFT,BETA,TAU(4),D(4,4),CC(4,4)
     1             ,XX(4),IFILL1(342)
      COMMON /ELPAR/ NPAR(14),IFILL2(10)
      DIMENSION     E(NUMTC,11,1),EE(10),C(4,4),P(4),ALP(4)
C
C          STRESS-STRAIN LAW IN  N-S-T  SYSTEM
C
       IF (NT.NE.1)  GO TO 220
      DO 210 KK=1,10
  210 EE(KK)=E(1,KK+1,MAT)
      GO TO 260
  220 DO 230 I=2,NT
      T1=E(I-1,1,MAT)
```

```
      T2=E(I,1,MAT)
      IF(T2.GE.TEMP) GO TO 240
  230 CONTINUE
  240 CONTINUE
      RI=(T2-TEMP)/(T2-T1)
      RJ=(TEMP-T1)/(T2-T1)
      DO 250 KK=1,10
  250 EE(KK)=E(I-1,KK+1,MAT)*RI+E(I,KK+1,MAT)*RJ
  260 CONTINUE
      DO 265 II=1,4
      DO 265 KK=1,4
      C(II,KK)=0.
  265 D(II,KK)=0.
C
      C(1,1)= 1.0/ EE(1)
      C(2,2)= 1.0/ EE(2)
      C(3,3)= 1.0/ EE(3)
      C(1,2)= -EE(4)/EE(2)
      C(1,3)= -EE(5)/EE(3)
      C(2,3)= -EE(6)/EE(3)
      C(2,1)= C(1,2)
      C(3,1)= C(1,3)
      C(3,2)= C(2,3)
      C(4,4)= 1.0/ EE(7)
C
      DO 270 M=1,3
      ALP(M)  = EE(M+7)
  270 CONTINUE
      ALP(4)  = 0.0
C
C       ROTATE MATERIAL PROPERTIES TO R-Z-T SYSTEM
C
      IF(BETA.EQ.0.0) GO TO 500
      ANG=BETA/57.2957795
      SS=SIN(ANG)
      ACC=COS(ANG)
      S2=SS*SS
      C2=ACC*ACC
      SC=SS*ACC
C     SET  D  FOR  SIG(O)=D*SIG(G)
      D(1,1)=C2
      D(1,2)=S2
      D(1,4)=2.*SC
      D(2,1)=S2
      D(2,2)=C2
      D(2,4)=-D(1,4)
      D(3,3)=1.0
      D(4,1)=-SC
      D(4,2)=-D(4,1)
      D(4,4)=C2-S2
C
C     FORM  (D)TRANSPOSE * (C)
C
      DO 300 I=1,4
      DO 300 J=1,4
```

```
      SUM=0.
      DO 280 M=1,4
  280 SUM=SUM+D(M,I)*C(M,J)
  300 CC(I,J)=SUM
C
C     FORM  (D)TRANSPOSE * (C) * (D)
C
      DO 350 I=1,4
      DO 350 J=1,4
      SUM=0.
      DO 330 M=1,4
  330 SUM=SUM+CC(I,M)*D(M,J)
      C(I,J)=SUM
  350 C(J,I)=SUM
C
C     TRANSFORM THERMAL EXPANSION COEFFICIENTS
C
      XX(1)=C2*ALP(1)+S2*ALP(2)
      XX(2)=S2*ALP(1)+C2*ALP(2)
      XX(3)=ALP(3)
      XX(4)=2.*SC*(ALP(1)-ALP(2))
      DO 430 I=1,4
  430 ALP(I) = XX(I)
C
C     INVERT THE STRAIN-STRESS LAW
C
  500 CALL POSINV (C,4,4)
C
C     MODIFY FOR THE CONDITION OF PLANE STRESS
C
      IF(NPAR(5).NE.2) GO TO 660
C
      C(1,1)= C(1,1)- C(3,1)* C(1,3)/C(3,3)
      C(1,2)= C(1,2)- C(3,2)* C(1,3)/C(3,3)
      C(1,4)= C(1,4)- C(3,4)* C(1,3)/C(3,3)
      C(2,2)= C(2,2)- C(3,2)* C(2,3)/C(3,3)
      C(2,4)= C(2,4)- C(3,4)* C(2,3)/C(3,3)
      C(4,4)= C(4,4)- C(3,4)* C(4,3)/C(3,3)
C
      DO 650 I=1,4
      DO 600 J=1,4
  600 C(J,I)=C(I,J)
      C(I,3)=0.
  650 C(3,I)=0.
C
C     RESTRAINED THERMAL STRESSES
C
  660 DO 670 I=1,4
      P(I) = 0.
      DO 670 M=1,4
  670 P(I)=P(I)+C(I,M)*ALP(M)
C
  700 RETURN
      END
      SUBROUTINE ELOUTH (KD,L,IELT,M,NS)
```

```
C
C     CALLED BY?  DISPLY
C
C     KD(1,I)   =   ELEMENT NUMBERS IN THIS OUTPUT SET
C                   (I RANGES FROM 1 TO L)
C     KD(2,I)   =   ELEMENT COMPONENT NUMBERS
C     L         =   NUMBER OF ELEMENT COMPONENT NUMBERS PER LINE OF
C                   OUTPUT  (8 MAXIMUM)
C     IELT      =   ELEMENT TYPE  (1,2,...,12)
C     M         =   OUTPUT SET NUMBER
C     NS        =   MAXIMUM NUMBER OF STRESS COMPONENTS ASSOCIATED
C                   WITH THE IELT-TH ELEMENT TYPE
C
      DIMENSION KD(3,1)
      DIMENSION SY(12,6),SZ(12,7),LAB(12),HD(12,4),HH(8,2)
C
C     ELEMENT COMPONENT LABELS
C
      DATA SY( 1,1) /3H   /
      DATA SY( 2,1) /3HP1(/, SY( 2,2) /3HV2(/, SY( 2,3) /3HV3(/,
     1     SY( 2,4) /3HT1(/, SY( 2,5) /3HM2(/, SY( 2,6) /3HM3(/
      DATA SY( 3,1) /3HPX(/, SY( 3,2) /3HVY(/, SY( 3,3) /3HVZ(/,
     1     SY( 3,4) /3HTX(/, SY( 3,5) /3HMY(/, SY( 3,6) /3HMZ(/
      DATA SY( 4,1) /3HV -/, SY( 4,2) /3HU -/, SY( 4,3) /3HT -/,
     1     SY( 4,4) /3HUV-/
      DATA SY( 5,1) /3HXX-/, SY( 5,2) /3HYY-/, SY( 5,3) /3HZZ-/,
     1     SY( 5,4) /3HXY-/, SY( 5,5) /3HYZ-/, SY( 5,6) /3HZX-/
      DATA SY( 6,1) /3HXX-/, SY( 6,2) /3HYY-/, SY( 6,3) /3HXY-/
      DATA SY( 7,1) /3HBDR/
      DATA SY( 8,1) /3HSXX/, SY( 8,2) /3HSYY/, SY( 8,3) /3HSZZ/,
     1     SY( 8,4) /3HSXY/, SY( 8,5) /3HSYZ/, SY( 8,6) /3HSZX/
      DATA SY(12,1) /3HPX(/, SY(12,2) /3HVY(/, SY(12,3) /3HVZ(/,
     1     SY(12,4) /3HTX(/, SY(12,5) /3HMY(/, SY(12,6) /3HMZ(/
C
      DATA SZ( 1,1) /3HP/A/, SZ( 1,2) /3HP  /
      DATA SZ( 2,1) /3HI) /, SZ( 2,2) /3HJ) /
      DATA SZ( 3,1) /3HI) /, SZ( 3,2) /3HJ) /
      DATA SZ( 4,1) /3HS0 /, SZ( 4,2) /3HS1 /, SZ( 4,3) /3HS2 /,
     1     SZ( 4,4) /3HS3 /, SZ( 4,5) /3HS4 /
      DATA SZ( 5,1) /3HSL1/, SZ( 5,2) /3HSL2/
      DATA SZ( 6,1) /3HS/R/, SZ( 6,2) /3HM/R/
      DATA SZ( 7,1) /3HY-F/, SZ( 7,2) /3HY-M/
      DATA SZ( 8,1) /3H(0)/, SZ( 8,2) /3H(1)/, SZ( 8,3) /3H(2)/,
     1     SZ( 8,4) /3H(3)/, SZ( 8,5) /3H(4)/, SZ( 8,6) /3H(5)/,
     2     SZ( 8,7) /3H(6)/
      DATA SZ(12,1) /3HI) /, SZ(12,2) /3HC) /, SZ(12,3) /3HJ) /
C
      DATA LAB /1,6,6,4,6,3,1,6,0,0,0,6/
C
C     ELEMENT TYPE LABELS
C
      DATA HD( 1,1)/6HT R U /,HD( 1,2)/6HS S   /,HD( 1,3)/6H      /
      DATA HD( 2,1)/6HB E A /,HD( 2,2)/6HM     /,HD( 2,3)/6H      /
      DATA HD( 3,1)/6H2/D   /,HD( 3,2)/6HP L A /,HD( 3,3)/6HN A R /
      DATA HD( 4,1)/6HA X I /,HD( 4,2)/6HS Y M /,HD( 4,3)/6HM E T /
```

```
      DATA HD( 5,1)/6H3/D   /,HD( 5,2)/6HB R I /,HD( 5,3)/6HC K   /
      DATA HD( 6,1)/6HP L A /,HD( 6,2)/6HT E / /,HD( 6,3)/6HS H E /
      DATA HD( 7,1)/6HB O U /,HD( 7,2)/6HN D A /,HD( 7,3)/6HR Y   /
      DATA HD( 8,1)/6HT H I /,HD( 8,2)/6HC K   /,HD( 8,3)/6HS H E /
      DATA HD(12,1)/6H3/D   /,HD(12,2)/6HP I P /,HD(12,3)/6HE     /
C
      DATA HD( 1,4)/6H      /
      DATA HD( 2,4)/6H      /
      DATA HD( 3,4)/6H      /
      DATA HD( 4,4)/6HR I C /
      DATA HD( 5,4)/6H      /
      DATA HD( 6,4)/6HL L   /
      DATA HD( 7,4)/6H      /
      DATA HD( 8,4)/6HL L   /
      DATA HD(12,4)/6H      /
C
C     DETERMINE ADJUSTED ELEMENT TYPE FOR TABLE SELECTION
C
      IF(L.LT.1) RETURN
      KEL = IELT
      IF(IELT.EQ.12 .AND. NS.EQ.12) KEL = 3
      IF(IELT.EQ.3) KEL = 4
      IF(IELT.EQ. 9) RETURN
      IF(IELT.EQ.10) RETURN
      IF(IELT.EQ.11) RETURN
C
C     TITLE PAGE WITH ELEMENT TYPE
C
      WRITE (6,2000)
 2000 FORMAT (42H1T I M E   H I S T O R Y   R E S P O N S E, / 1X)
      WRITE (6,2010) (HD(IELT,K),K=1,4), M
 2010 FORMAT (15H ELEMENT TYPE (,4A6,24H)   / / /   OUTPUT SET =,I4/ 1X)
      WRITE (6,2020)
 2020 FORMAT (14X,37H*ELEMENT NUMBER* - *STRESS COMPONENT*,  1X)
C
C     SELECT THE LABEL INDEX FOR THIS ELEMENT TYPE
C
      N1 = LAB(KEL)
C
C     SELECT ELEMENT COMPONENT HEADINGS
C
      DO 10 N=1,L
      J = (KD(2,N)-1)/ N1 + 1
      HH(N,2) = SZ(KEL,J)
      J = KD(2,N) - (J-1)* N1
      HH(N,1) = SY(KEL,J)
   10 CONTINUE
C
C     WRITE THE HEADING LINE
C
      WRITE (6,2030) (KD(1,I),HH(I,1),HH(I,2),I=1,L)
 2030 FORMAT (8X,4HTIME,2X, 8(I5,1H-,2A3), / 1X)
C
      RETURN
      END
```

```
      SUBROUTINE ELOUTR (NEL,IS,L,IELT,NS)
C
C     CALLED BY?  STRESR
C
C     NEL     =   ELEMENT NUMBER
C     IS      =   ELEMENT COMPONENT NUMBERS
C     L       =   NUMBER OF ELEMENT COMPONENT NUMBERS PER LINE OF
C                 OUTPUT (12 MAXIMUM)
C     IELT    =   ELEMENT TYPE  (1,2,...,12)
C     NS      =   MAXIMUM NUMBER OF STRESS COMPONENTS ASSOCIATED
C                 WITH THE IELT-TH ELEMENT TYPE
C
      DIMENSION IS(1)
      DIMENSION SY(12,6),SZ(12,7),LAB(12),HD(12,4),HH(12,2)
C
C     ELEMENT COMPONENT LABELS
C
      DATA SY( 1,1) /3H   /
      DATA SY( 2,1) /3HP1(/, SY( 2,2) /3HV2(/, SY( 2,3) /3HV3(/,
     1     SY( 2,4) /3HT1(/, SY( 2,5) /3HM2(/, SY( 2,6) /3HM3(/
      DATA SY( 3,1) /3HPX(/, SY( 3,2) /3HVY(/, SY( 3,3) /3HVZ(/,
     1     SY( 3,4) /3HTX(/, SY( 3,5) /3HMY(/, SY( 3,6) /3HMZ(/
      DATA SY( 4,1) /3HV -/, SY( 4,2) /3HU -/, SY( 4,3) /3HT -/,
     1     SY( 4,4) /3HUV-/
      DATA SY( 5,1) /3HXX-/, SY( 5,2) /3HYY-/, SY( 5,3) /3HZZ-/,
     1     SY( 5,4) /3HXY-/, SY( 5,5) /3HYZ-/, SY( 5,6) /3HZX-/
      DATA SY( 6,1) /3HXX-/, SY( 6,2) /3HYY-/, SY( 6,3) /3HXY-/
      DATA SY( 7,1) /3HBDR/
      DATA SY( 8,1) /3HSXX/, SY( 8,2) /3HSYY/, SY( 8,3) /3HSZZ/,
     1     SY( 8,4) /3HSXY/, SY( 8,5) /3HSYZ/, SY( 8,6) /3HSZX/
      DATA SY(12,1) /3HPX(/, SY(12,2) /3HVY(/, SY(12,3) /3HVZ(/,
     1     SY(12,4) /3HTX(/, SY(12,5) /3HMY(/, SY(12,6) /3HMZ(/
C
      DATA SZ( 1,1) /3HP/A/, SZ( 1,2) /3HP  /
      DATA SZ( 2,1) /3HI) /, SZ( 2,2) /3HJ) /
      DATA SZ( 3,1) /3HI) /, SZ( 3,2) /3HJ) /
      DATA SZ( 4,1) /3HS0 /, SZ( 4,2) /3HS1 /, SZ( 4,3) /3HS2 /,
     1     SZ( 4,4) /3HS3 /, SZ( 4,5) /3HS4 /
      DATA SZ( 5,1) /3HSL1/, SZ( 5,2) /3HSL2/
      DATA SZ( 6,1) /3HS/R/, SZ( 6,2) /3HM/R/
      DATA SZ( 7,1) /3HY-F/, SZ( 7,2) /3HY-M/
      DATA SZ( 8,1) /3H(0)/, SZ( 8,2) /3H(1)/, SZ( 8,3) /3H(2)/,
     1     SZ( 8,4) /3H(3)/, SZ( 8,5) /3H(4)/, SZ( 8,6) /3H(5)/,
     2     SZ( 8,7) /3H(6)/
      DATA SZ(12,1) /3HI) /, SZ(12,2) /3HC) /, SZ(12,3) /3HJ) /
C
      DATA LAB /1,6,6,4,6,3,1,6,0,0,0,6/
C
C     ELEMENT TYPE LABELS
C
      DATA HD( 1,1)/6HT R U /,HD( 1,2)/6HS S   /,HD( 1,3)/6H      /
      DATA HD( 2,1)/6HB E A /,HD( 2,2)/6HM     /,HD( 2,3)/6H      /
      DATA HD( 3,1)/6H2/D   /,HD( 3,2)/6HP L A /,HD( 3,3)/6HN A R /
      DATA HD( 4,1)/6HA X I /,HD( 4,2)/6HS Y M /,HD( 4,3)/6HM E T /
      DATA HD( 5,1)/6H3/D   /,HD( 5,2)/6HB R I /,HD( 5,3)/6HC K   /
```

```
      DATA HD( 6,1)/6HP L A /,HD( 6,2)/6HT E / /,HD( 6,3)/6HS H E /
      DATA HD( 7,1)/6HB O U /,HD( 7,2)/6HN D A /,HD( 7,3)/6HR Y   /
      DATA HD( 8,1)/6HT H I /,HD( 8,2)/6HC K   /,HD( 8,3)/6HS H E /
      DATA HD(12,1)/6H3/D   /,HD(12,2)/6HP I P /,HD(12,3)/6HE     /
C
      DATA HD( 1,4)/6H      /
      DATA HD( 2,4)/6H      /
      DATA HD( 3,4)/6H      /
      DATA HD( 4,4)/6HR I C /
      DATA HD( 5,4)/6H      /
      DATA HD( 6,4)/6HL L   /
      DATA HD( 7,4)/6H      /
      DATA HD( 8,4)/6HL L   /
      DATA HD(12,4)/6H      /
C
C     DETERMINE ADJUSTED ELEMENT TYPE FOR TABLE SELECTION
C
      IF(L.LT.1) RETURN
      KEL = IELT
      IF(IELT.EQ.12 .AND. NS.EQ.12) KEL = 3
      IF(IELT.EQ.3) KEL = 4
      IF(IELT.EQ. 9) RETURN
      IF(IELT.EQ.10) RETURN
      IF(IELT.EQ.11) RETURN
C
C     TITLE PAGE WITH ELEMENT TYPE
C
      WRITE (6,2010) (HD(IELT,K),K=1,4), NEL
 2010 FORMAT (15HOELEMENT TYPE (,4A6,28H)   / / /   ELEMENT NUMBER (,
     1         I4, 1H), / 1X)
C
C     SELECT ELEMENT COMPONENT HEADINGS
C
      N1 = LAB(KEL)
      DO 10 N=1,L
      J = (IS(N)  -1)/ N1 + 1
      HH(N,2) = SZ(KEL,J)
      J = IS(N)   - (J-1)* N1
      HH(N,1) = SY(KEL,J)
   10 CONTINUE
C
C     WRITE THE HEADING LINE
C
      WRITE (6,2030)         (HH(I,1),HH(I,2),I=1,L)
 2030 FORMAT (12(5X,2A3) )
C
      RETURN
      END
      SUBROUTINE ELOUTS (KD,L,IELT,M,NS)
C
C     CALLED BY?  SDSPLY
C
C     KD(1,I)  =  ELEMENT NUMBERS IN THIS OUTPUT SET
C                 (I RANGES FROM 1 TO L)
C     KD(2,I)  =  ELEMENT COMPONENT NUMBERS
```

```
C     L          =    NUMBER OF ELEMENT COMPONENT NUMBERS PER LINE OF
C                     OUTPUT   (8 MAXIMUM)
C     IELT       =    ELEMENT TYPE   (1,2,...,12)
C     M          =    OUTPUT SET NUMBER
C     NS         =    MAXIMUM NUMBER OF STRESS COMPONENTS ASSOCIATED
C                     WITH THE IELT-TH ELEMENT TYPE
C
      DIMENSION KD(2,1)
      DIMENSION SY(12,6),SZ(12,7),LAB(12),HD(12,4),HH(8,2)
C
C     ELEMENT COMPONENT LABELS
C
      DATA SY( 1,1) /3H   /
      DATA SY( 2,1) /3HP1(/, SY( 2,2) /3HV2(/, SY( 2,3) /3HV3(/,
     1     SY( 2,4) /3HT1(/, SY( 2,5) /3HM2(/, SY( 2,6) /3HM3(/
      DATA SY( 3,1) /3HPX(/, SY( 3,2) /3HVY(/, SY( 3,3) /3HVZ(/,
     1     SY( 3,4) /3HTX(/, SY( 3,5) /3HMY(/, SY( 3,6) /3HMZ(/
      DATA SY( 4,1) /3HV -/, SY( 4,2) /3HU -/, SY( 4,3) /3HT -/,
     1     SY( 4,4) /3HUV-/
      DATA SY( 5,1) /3HXX-/, SY( 5,2) /3HYY-/, SY( 5,3) /3HZZ-/,
     1     SY( 5,4) /3HXY-/, SY( 5,5) /3HYZ-/, SY( 5,6) /3HZX-/
      DATA SY( 6,1) /3HXX-/, SY( 6,2) /3HYY-/, SY( 6,3) /3HXY-/
      DATA SY( 7,1) /3HBDR/
      DATA SY( 8,1) /3HSXX/, SY( 8,2) /3HSYY/, SY( 8,3) /3HSZZ/,
     1     SY( 8,4) /3HSXY/, SY( 8,5) /3HSYZ/, SY( 8,6) /3HSZX/
      DATA SY(12,1) /3HPX(/, SY(12,2) /3HVY(/, SY(12,3) /3HVZ(/,
     1     SY(12,4) /3HTX(/, SY(12,5) /3HMY(/, SY(12,6) /3HMZ(/
C
      DATA SZ( 1,1) /3HP/A/, SZ( 1,2) /3HP  /
      DATA SZ( 2,1) /3HI) /, SZ( 2,2) /3HJ) /
      DATA SZ( 3,1) /3HI) /, SZ( 3,2) /3HJ) /
      DATA SZ( 4,1) /3HS0 /, SZ( 4,2) /3HS1 /, SZ( 4,3) /3HS2 /,
     1     SZ( 4,4) /3HS3 /, SZ( 4,5) /3HS4 /
      DATA SZ( 5,1) /3HSL1/, SZ( 5,2) /3HSL2/
      DATA SZ( 6,1) /3HS/R/, SZ( 6,2) /3HM/R/
      DATA SZ( 7,1) /3HY-F/, SZ( 7,2) /3HY-M/
      DATA SZ( 8,1) /3H(0)/, SZ( 8,2) /3H(1)/, SZ( 8,3) /3H(2)/,
     1     SZ( 8,4) /3H(3)/, SZ( 8,5) /3H(4)/, SZ( 8,6) /3H(5)/,
     2     SZ( 8,7) /3H(6)/
      DATA SZ(12,1) /3HI) /, SZ(12,2) /3HC) /, SZ(12,3) /3HJ) /
C
      DATA LAB /1,6,6,4,6,3,1,6,0,0,0,6/
C
C     ELEMENT TYPE LABELS
C
      DATA HD( 1,1)/6HT R U /,HD( 1,2)/6HS S   /,HD( 1,3)/6H      /
      DATA HD( 2,1)/6HB E A /,HD( 2,2)/6HM     /,HD( 2,3)/6H      /
      DATA HD( 3,1)/6H2/D   /,HD( 3,2)/6HP L A /,HD( 3,3)/6HN A R /
      DATA HD( 4,1)/6HA X I /,HD( 4,2)/6HS Y M /,HD( 4,3)/6HM E T /
      DATA HD( 5,1)/6H3/D   /,HD( 5,2)/6HB R I /,HD( 5,3)/6HC K   /
      DATA HD( 6,1)/6HP L A /,HD( 6,2)/6HT E / /,HD( 6,3)/6HS H E /
      DATA HD( 7,1)/6HB O U /,HD( 7,2)/6HN D A /,HD( 7,3)/6HR Y   /
      DATA HD( 8,1)/6HT H I /,HD( 8,2)/6HC K   /,HD( 8,3)/6HS H E /
      DATA HD(12,1)/6H3/D   /,HD(12,2)/6HP I P /,HD(12,3)/6HE     /
C
```

```
      DATA HD( 1,4)/6H      /
      DATA HD( 2,4)/6H      /
      DATA HD( 3,4)/6H      /
      DATA HD( 4,4)/6HR I C /
      DATA HD( 5,4)/6H      /
      DATA HD( 6,4)/6HL L   /
      DATA HD( 7,4)/6H      /
      DATA HD( 8,4)/6HL L   /
      DATA HD(12,4)/6H      /
C
C     DETERMINE ADJUSTED ELEMENT TYPE FOR TABLE SELECTION
C
      IF(L.LT.1) RETURN
      KEL = IELT
      IF(IELT.EQ.12 .AND. NS.EQ.12) KEL = 3
      IF(IELT.EQ.3) KEL = 4
      IF(IELT.EQ. 9) RETURN
      IF(IELT.EQ.10) RETURN
      IF(IELT.EQ.11) RETURN
C
C     TITLE PAGE WITH ELEMENT TYPE
C
      WRITE (6,2000)
 2000 FORMAT (42H1T I M E   H I S T O R Y   R E S P O N S E, /  1X)
      WRITE (6,2010) (HD(IELT,K),K=1,4), M
 2010 FORMAT (15H ELEMENT TYPE (,4A6,24H)   / / /   OUTPUT SET =,I4/ 1X)
      WRITE (6,2020)
 2020 FORMAT (13X,40H *ELEMENT NUMBER* - (*STRESS COMPONENT*), 1X)
C
C     SELECT THE LABEL INDEX FOR THIS ELEMENT TYPE
C
      N1 = LAB(KEL)
C
C     SELECT ELEMENT COMPONENT HEADINGS
C
      DO 10 N=1,L
      J = (KD(2,N)-1)/ N1 + 1
      HH(N,2) = SZ(KEL,J)
      J = KD(2,N) - (J-1)* N1
      HH(N,1) = SY(KEL,J)
   10 CONTINUE
C
C     WRITE THE HEADING LINE
C
      WRITE (6,2030) (KD(1,I),HH(I,1),HH(I,2),I=1,L)
 2030 FORMAT (8X, 4HTIME,2X, 8(I5,1H-,2A3) )
C
      RETURN
      END
      SUBROUTINE EMID (ID,MASS,NUMNP,NEQB)
C
C     CALLED BY?  HISTRY
C
      DIMENSION ID(NUMNP,6),MASS(NEQB)
C
```

```
      REWIND 3
      REWIND 8
      READ (8) ID
      L=1
      DO 200 N=1,NUMNP
      DO 100 I=1,6
   50 MASS(L)=0
      IF(ID(N,I).LE.0) GO TO 100
      IF(L.LE.NEQB) GO TO 75
      WRITE (3) MASS
      L=1
   75 IF(I.GT.3) GO TO 90
      MASS(L)=I
   90 L=L+1
  100 CONTINUE
  200 CONTINUE
      DO 300 I=L,NEQB
  300 MASS(I)=0
      WRITE (3) MASS
C
      RETURN
      END
      SUBROUTINE EMIDR (ID,MASS,NUMNP,NEQB)
C
C     CALLED BY?  RESPEC
C
      DIMENSION ID(NUMNP,6),MASS(NEQB)
C
      REWIND 3
      REWIND 8
      READ (8) ID
      L=1
      DO 200 N=1,NUMNP
      DO 100 I=1,6
   50 MASS(L)=0
      IF(ID(N,I).LE.0) GO TO 100
      IF(L.LE.NEQB) GO TO 75
      WRITE (3) MASS
      L=1
   75 IF(I.GT.3) GO TO 90
      MASS(L)=I
   90 L=L+1
  100 CONTINUE
  200 CONTINUE
      DO 300 I=L,NEQB
  300 MASS(I)=0
      WRITE (3) MASS
C
      RETURN
      END
      SUBROUTINE EMIDS(ID,MASS,NUMNP,NEQ)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE CREATES AN INTEGER ARRAY *MASS* WHICH FLAGS THE
```

```
C     TRANSLATIONAL COMPONENT NUMBERS (1=X,2=Y,3=Z) ASSOCIATED WITH
C     EACH SYSTEM DEGREE OF FREEDOM.  *MASS* IS SAVED ON TAPE7 FOR
C     LATER USE IN SUBROUTINE *GROUND*.  *MASS* ELEMENTS EQ.0 INDICATE
C     ROTATIONAL COMPONENT FOR THAT DEGREE OF FREEDOM.
C
      DIMENSION  ID(NUMNP,6),MASS(NEQ)
C
      NT=7
      REWIND NT
C
      L=1
      DO 200 N=1,NUMNP
      DO 100 I=1,6
   50 MASS(L)=0
      IF(ID(N,I).LE.0)  GO TO 100
      IF(I.GT.3)  GO TO 90
      MASS(L)=I
   90 L=L+1
  100 CONTINUE
  200 CONTINUE
C
      WRITE (NT) MASS
      RETURN
      END
      SUBROUTINE FACEPR (NEL,KDIS,KXYZ,XX,NOD9,H,P,PL,NFACE,LT,PWA,KLS)
C
C     CALLED BY ? THDFE
C     CALLS ? FNCT
C
C
C
C     THIS ROUTINE COMPUTES NODE FORCES DUE TO APPLIED ELEMENT FACE
C     PRESSURE DISTRIBUTIONS
C
C
      DIMENSION    XX(3,1),NOD9(1),H(1),P(3,1),PL(1),PWA(1)
      DIMENSION    XJ(3,3),ETA(3),KFACE(6,8),KCRD(6),FVAL(6),IPRM(3),
     1             PR(8),NODES(8),IPR4(4)
      COMMON /GAUSS/ XK(4,4),WGT(4,4)
C
      DATA KFACE / 1, 2, 1, 4, 1, 5,
     1              4, 3, 5, 8, 2, 6,
     2              8, 7, 6, 7, 3, 7,
     3              5, 6, 2, 3, 4, 8,
     4             12, 10, 17, 20,  9, 13,
     5             20, 19, 13, 15, 10, 14,
     6             16, 14, 18, 19, 11, 15,
     7             17, 18,  9, 11, 12, 16/
C
      DATA  KCRD  / 1,  1,  2,  2,  3,  3/
      DATA  FVAL  / 1.,-1., 1.,-1., 1.,-1./
      DATA  IPRM  /  2,  3,  1/
      DATA IPR4  /  2,  3,  4,  1/
C
C     DETERMINE THE ELEMENT NODES CONTRIBUTING TO FORCE CALCULATIONS
```

```
C      ON THIS FACE
C
       DO 2 I=1,4
       NODES(I ) = KFACE(NFACE,I)
       NODES(I+4) = 0
     2 CONTINUE
C
       IF(KDIS.LT.9) GO TO 9
C
       NN9 = KDIS-8
C
       DO 8 K=5,8
       DO 4 I=1,NN9
C
       J = I
       IF(KFACE(NFACE,K).EQ.NOD9(I)) GO TO 6
C
     4 CONTINUE
       GO TO 8
C
     6 NODES(K) = J
     8 CONTINUE
C
     9 CONTINUE
C
C      SET UP THE PRESSURE VECTOR FOR THE FOUR FACE CORNER NODES
C
C         1. ADJUST THE SIGN OF THE PRESSURES SO THAT POSITIVE
C            PRESSURE ALWAYS COMPRESSES THE ELEMENT
C
       FACT = -FVAL(NFACE)
C
       GO TO (10,30), LT
C
C         2. DISTRIBUTED PRESSURE GIVEN AT THE CORNER NODES
C
    10 DO 25 K=1,8
C
       IF(NODES(K).EQ.0) GO TO 25
C
       IF(K.GT.4) GO TO 15
C
       PR(K) = PWA(K) * FACT
       GO TO 25
C
    15 J = K-4
       L = IPR4(J)
       PR(K) = (PWA(J) + PWA(L)) * 0.5 * FACT
C
    25 CONTINUE
       GO TO 75
C
C         3. ELEMENT FACE EXPOSED TO HYDROSTATIC PRESSURE
C
    30 GAMMA = PWA(1)* FACT
```

```
C
      XLN = 0.0
      DO 35 K=1,3
      ETA(K) =  PWA(K+4) - PWA(K+1)
   35 XLN = XLN + ETA(K)**2
      XLN =SQRT(XLN)
C
      IF(XLN.GT.1.0E-6) GO TO 40
C
      WRITE (6,3000) KLS,NEL
 3000 FORMAT (31HOERROR***  PRESSURE LOAD SET (,I3,15H) FOR ELEMENT (,
     1          I5,43H) HAS UNDEFINED HYDROSTATIC SURFACE NORMAL., / 1X)
      STOP
C
   40 DO 45 K=1,3
   45 ETA(K) = ETA(K)/ XLN
C
      DO 70 N=1,8
C
      IF(NODES(N).EQ.0) GO TO 70
C
      XLN = 0.0
               NOD = NODES(N)
      IF(N.GT.4) NOD = NOD + 8
C
      DO 50 I=1,3
   50 XLN = XLN + (XX(I,NOD) - PWA(I+1))* ETA(I)
C
      PR(N) = XLN* GAMMA
C
      IF(XLN.LT.0.0) PR(N) = 0.0
C
   70 CONTINUE
   75 CONTINUE
C
C     SET UP VARIABLES FOR THE SURFACE INTEGRATION
C
      ML = KCRD(NFACE)
      MM = IPRM(ML)
      MN = IPRM(MM)
C
C     SURFACE INTEGRATION LOOP
C
      ETA(ML) = FVAL(NFACE)
C
      DO 300 LX=1,3
C
      ETA(MM) = XK(LX,3)
C
      DO 300 LY=1,3
C
      ETA(MN) = XK(LY,3)
C
      WT = WGT(LX,3)* WGT(LY,3)
C
```

```
C     EVALUATE THE INTERPOLATION FUNCTIONS AND JACOBIAN MATRIX
C
      CALL FNCT (ETA(1),ETA(2),ETA(3),H,P,NOD9,XJ,DET,XX,KDIS,KXYZ,NEL)
C
C     COMPUTE THE DIRECTION COSINES OF THE UNIT SURFACE NORMAL VECTOR
C     AT THIS SAMPLE POINT
C
      A1 = XJ(MM,2)* XJ(MN,3)- XJ(MM,3)* XJ(MN,2)
      A2 = XJ(MM,3)* XJ(MN,1)- XJ(MM,1)* XJ(MN,3)
      A3 = XJ(MM,1)* XJ(MN,2)- XJ(MM,2)* XJ(MN,1)
C
      AA =SQRT(A1**2 + A2**2 + A3**2)
      IF(AA.GT.1.0E-8) GO TO 100
C
      WRITE (6,3010) NFACE,NEL
 3010 FORMAT (38HOERROR***   UNDEFINED NORMAL IN FACE (,I1,5H) FOR,
     1          10H ELEMENT (,I5,2H)., / 1X)
      STOP
C
  100 FACT = 1.0/AA
      A1 = A1* FACT
      A2 = A2* FACT
      A3 = A3* FACT
C
C     COMPUTE THE FIRST FUNDAMENTAL FORM (AREA DIFFERENTIAL)
C
      AA = 0.0
      BB = 0.0
      CC = 0.0
      DO 120 I=1,3
      AA = AA + XJ(MM,I)**2
      CC = CC + XJ(MN,I)**2
  120 BB = BB + XJ(MM,I)* XJ(MN,I)
      C  =SQRT(AA*CC - BB**2)
C
C     INTERPOLATE FOR THE PRESSURE AT THIS SAMPLE POINT
C
      PRESS = 0.0
C
      DO 130 K=1,8
C
      IF(NODES(K).EQ.0) GO TO 130
C
                NOD = NODES(K)
      IF(K.GT.4) NOD = NOD + 8
C
      PRESS = PRESS + H(NOD)* PR(K)
  130 CONTINUE
C
      FACT = WT* C* PRESS
C
C     ASSEMBLE THE NODE FORCE CONTRIBUTION
C
      DO 160 L=1,8
C
```

```
      IF(NODES(L).EQ.0) GO TO 160
C
      IF(L.GT.4) GO TO 140
C
C        1. CORNER NODES
C
      N = NODES(L)
      K = 3*N
      GO TO 150
C
C        2. SIDE NODES
C
  140 J = NODES(L)
      N = J+8
      K = 3* NOD9(J)
C
  150 QQ = FACT* H(N)
C
      PL(K-2) = PL(K-2) + QQ* A1
      PL(K-1) = PL(K-1) + QQ* A2
      PL(K  ) = PL(K  ) + QQ* A3
  160 CONTINUE
C
  300 CONTINUE
C
      RETURN
      END
      SUBROUTINE FNCT (R,S,T,H,P,NOD9,XJ,DET,XX,IELD,IELX,NEL)
C
C     CALLED BY ? FACEPR
C
C
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                                    .
C .   P R O G R A M                                                    .
C .                                                                    .
C .      TO FIND INTERPOLATION FUNCTIONS ( H )                         .
C .      AND DERIVATIVES ( P ) CORRESPONDING TO THE NODAL              .
C .      POINTS OF A CURVILINEAR ISOPARAMETRIC HEXAHEDRON              .
C .      OR SUBPARAMETRIC HEXAHEDRON (8 TO 21 NODES)                   .
C .                                                                    .
C .      TO FIND JACOBIAN ( XJ ) AND ITS DETERMINANT ( DET )           .
C .                                                                    .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C
      DIMENSION H(1),P(3,1),NOD9(1),IPERM(8),XJ(3,3),XX(3,1)
C
      DATA IPERM / 2,3,4,1,6,7,8,5 /
C
      IEL = IELD
      NND9= IELD-8
C
      RP=1.0 + R
```

```
      SP=1.0 + S
      TP=1.0 + T
      RM=1.0 - R
      SM=1.0 - S
      TM=1.0 - T
      RR=1.0 - R*R
      SS=1.0 - S*S
      TT=1.0 - T*T
C
C
C     INTERPOLATION FUNCTIONS AND THEIR DERIVATIVES
C
C
C     8-NODE BRICK
C
      H(1)=0.125*RP*SP*TP
      H(2) =0.125*RM*SP*TP
      H(3)=0.125*RM*SM*TP
      H(4)=0.125*RP*SM*TP
      H(5)=0.125*RP*SP*TM
      H(6)=0.125*RM*SP*TM
      H(7)=0.125*RM*SM*TM
      H(8)=0.125*RP*SM*TM
C
      P(1,1)=0.125*SP*TP
      P(1,2)=-P(1,1)
      P(1,3)=-0.125*SM*TP
      P(1,4)=-P(1,3)
      P(1,5)=0.125*SP*TM
      P(1,6)=-P(1,5)
      P(1,7)=-0.125*SM*TM
      P(1,8)=-P(1,7)
C
      P(2,1)=0.125*RP*TP
      P(2,2)=0.125*RM*TP
      P(2,3)=-P(2,2)
      P(2,4)=-P(2,1)
      P(2,5)=0.125*RP*TM
      P(2,6)=0.125*RM*TM
      P(2,7)=-P(2,6)
      P(2,8)=-P(2,5)
C
      P(3,1)=0.125*RP*SP
      P(3,2)=0.125*RM*SP
      P(3,3)=0.125*RM*SM
      P(3,4)=0.125*RP*SM
      P(3,5)=-P(3,1)
      P(3,6)=-P(3,2)
      P(3,7)=-P(3,3)
      P(3,8)=-P(3,4)
C
      IF(IEL.EQ.8) GO TO 50
C
C
C     ADD DEGREES OF FREEDOM IN EXCESS OF 8
```

```
C
      I=0
    2 I=I + 1
      IF(I.GT.NND9) GO TO 40
      NN=NOD9(I) - 8
      GO TO (9,10,11,12,13,14,15,16,17,18,19,20,21) ,NN
C
    9 H(9)  = 0.25*RR*SP*TP
      P(1,9)  = -0.50*R*SP*TP
      P(2,9)  = 0.25*RR*TP
      P(3,9)  = 0.25*RR*SP
      GO TO 2
   10 H(10)=0.25*RM*SS*TP
      P(1,10)=-0.25*SS*TP
      P(2,10)=-0.50*RM*S*TP
      P(3,10)= 0.25*RM*SS
      GO TO 2
   11 H(11)=0.25*RR*SM*TP
      P(1,11)=-0.50*R*SM*TP
      P(2,11)=-0.25*RR*TP
      P(3,11)= 0.25*RR*SM
      GO TO 2
   12 H(12)=0.25*RP*SS*TP
      P(1,12)= 0.25*SS*TP
      P(2,12)=-0.50*RP*S*TP
      P(3,12)= 0.25*RP*SS
      GO TO 2
   13 H(13)=0.25*RR*SP*TM
      P(1,13)=-0.50*R*SP*TM
      P(2,13)= 0.25*RR*TM
      P(3,13)=-0.25*RR*SP
      GO TO 2
   14 H(14)=0.25*RM*SS*TM
      P(1,14)=-0.25*SS*TM
      P(2,14)=-0.50*RM*S*TM
      P(3,14)=-0.25*RM*SS
      GO TO 2
   15 H(15)=0.25*RR*SM*TM
      P(1,15)=-0.50*R*SM*TM
      P(2,15)=-0.25*RR*TM
      P(3,15)=-0.25*RR*SM
      GO TO 2
   16 H(16)=0.25*RP*SS*TM
      P(1,16)= 0.25*SS*TM
      P(2,16)=-0.50*RP*S*TM
      P(3,16)=-0.25*RP*SS
      GO TO 2
   17 H(17)=0.25*RP*SP*TT
      P(1,17)=0.25*SP*TT
      P(2,17)=0.25*RP*TT
      P(3,17)=-0.50*RP*SP*T
      GO TO 2
   18 H(18)=0.25*RM*SP*TT
      P(1,18)=-0.25*SP*TT
      P(2,18)= 0.25*RM*TT
```

```
      P(3,18)=-0.50*RM*SP*T
      GO TO 2
   19 H(19)=0.25*RM*SM*TT
      P(1,19)=-0.25*SM*TT
      P(2,19)=-0.25*RM*TT
      P(3,19)=-0.50*RM*SM*T
      GO TO 2
   20 H(20)=0.25*RP*SM*TT
      P(1,20)= 0.25*SM*TT
      P(2,20)=-0.25*RP*TT
      P(3,20)=-0.50*RP*SM*T
      GO TO 2
   21 H(21)=RR*SS*TT
      P(1,21)=-2.0*R*SS*TT
      P(2,21)=-2.0*S*RR*TT
      P(3,21)=-2.0*T*RR*SS
      GO TO 2
C
C     MODIFT FIRST 8 FUNCTIONS IF 9 OR MORE NODES IN ELEMENT
C
   40 IH=0
   41 IH=IH + 1
      IF (IH.GT.NND9) GO TO 50
      II=IH + 7
      IF (II.EQ.IELX) GO TO 51
   42 IN=NOD9(IH)
      IF (IN.GT.16) GO TO 46
      I1=IN -8
      I2=IPERM(I1)
      H(I1)=H(I1)  - 0.5*H(IN)
      H(I2)=H(I2)  - 0.5*H(IN)
      H(IH+8)=H(IN)
      DO 45 J=1,3
      P(J,I1)=P(J,I1)  - 0.5*P(J,IN)
      P(J,I2)=P(J,I2)  - 0.5*P(J,IN)
   45 P(J,IH+8)=P(J,IN)
      GO TO 41
   46 IF (IN.EQ.21) GO TO 30
      I1=IN -16
      I2=I1 + 4
      H(I1)=H(I1)  - 0.5*H(IN)
      H(I2)=H(I2)  - 0.5*H(IN)
      H(IH+8)=H(IN)
      DO 47 J=1,3
      P(J,I1)=P(J,I1)  - 0.5*P(J,IN)
      P(J,I2)=P(J,I2)  - 0.5*P(J,IN)
   47 P(J,IH+8)=P(J,IN)
      GO TO 41
C
C     MODIFY FIRST 20 FUNCTIONS IF NODE 21 IS PRESENT
C
   30 IH=0
   31 IH=IH + 1
      IN=NOD9(IH)
      IF (IN.EQ.21) GO TO 35
```

```
      IF (IN.GT.16) GO TO 33
      I1=IN -8
      I2=IPERM(I1)
      H(I1)=H(I1) + 0.125*H(21)
      H(I2)=H(I2) + 0.125*H(21)
      DO 32 J=1,3
      P(J,I1)=P(J,I1) + 0.125*P(J,21)
   32 P(J,I2)=P(J,I2) + 0.125*P(J,21)
      GO TO 31
   33 I1=IN - 16
      I2=I1 + 4
      H(I1)=H(I1) + 0.125*H(21)
      H(I2)=H(I2) + 0.125*H(21)
      DO 34 J=1,3
      P(J,I1)=P(J,I1) + 0.125*P(J,21)
   34 P(J,I2)=P(J,I2) + 0.125*P(J,21)
      GO TO 31
   35 DO 36 I=1,8
      H(I)=H(I) - 0.125*H(21)
      DO 36 J=1,3
   36 P(J,I)=P(J,I) - 0.125*P(J,21)
      NN=NND9 + 7
      IF (NN.EQ.8) GO TO 50
      DO 38 I=9,NN
      H(I)=H(I) - 0.25*H(21)
      DO 38 J=1,3
   38 P(J,I)=P(J,I) - 0.25*P(J,21)
      H(NND9+8)=H(21)
      DO 39 J =1,3
   39 P(J,NND9+8)=P(J,21)
C
C
C     EVALUATE JACOBIAN MATRIX AT POINT (R,S,T)
C
C
   50 IF (IELX.LT.IELD) RETURN
   51 DO 100 I=1,3
      DO 100 J=1,3
      DUM=0.0
      DO 90 K=1,IELX
   90 DUM=DUM + P(I,K)*XX(J,K)
  100 XJ(I,J)=DUM
C
C
C     COMPUTE DETERMINANT OF JACOBIAN MATRIX AT POINT (R,S,T)
C
C
      DET = XJ(1,1)*XJ(2,2)*XJ(3,3)
     1    + XJ(1,2)*XJ(2,3)*XJ(3,1)
     2    + XJ(1,3)*XJ(2,1)*XJ(3,2)
     3    - XJ(1,3)*XJ(2,2)*XJ(3,1)
     4    - XJ(1,2)*XJ(2,1)*XJ(3,3)
     5    - XJ(1,1)*XJ(2,3)*XJ(3,2)
      IF(DET.GT.1.0E-8) GO TO 110
      WRITE (6,2000) NEL,R,S,T
```

```
      STOP
  110 IF (IELX.LT.IELD) GO TO 42
C
C
      RETURN
C
C
C
 2000 FORMAT (49HOERROR***   NEGATIVE OR ZERO JACOBIAN DETERMINANT,
     1        23H COMPUTED FOR ELEMENT (,I5,1H), /
     2        12X,  3HR =, F10.5 /
     3     12X,  3HS =, F10.5 /
     4        12X,  3HT =, F10.5 / 1X)
C
C
      END
      SUBROUTINE FORMB(S,T,B)
C
C     CALLED BY?  QUAD
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/    LM(12),U(12,12),P(12,4),XM(12),
     1 TI(20,4),IX(4),IE(5),NS,D(4,4),EMUL(4,5),RR(4),ZZ(4),H(6),HS(6),
     2 HT(6),HR(6),HZ(6),FAC,XMM,PRESS,    EE(10),TTI(4),PP(12,4),THICK
     3 ,TMP(4),QP(12),ALP(4),IFILL2(4236)
      DIMENSION B(20,12)
      DIMENSION II(6),JJ(6)
      DATA II/1,2,3,4,9,10/,JJ/5,6,7,8,11,12/
C
      SM=1.0-S
      SP=1.0+S
      TM=1.0-T
      TP=1.0+T
C
      H(1)=SM*TM/4.
      H(2)=SP*TM/4.
      H(3)=SP*TP/4.
      H(4)=SM*TP/4.
      H(5)=(1.0-S*S)
      H(6)=(1.0-T*T)
C
      HS(1)=-TM/4.
      HS(2)=-HS(1)
      HS(3)=TP/4.
      HS(4)=-HS(3)
      HS(5)=-2.*S
      HS(6)=0.0
C
      HT(1)=-SM/4.
      HT(2)=-SP/4.
      HT(3)=-HT(2)
      HT(4)=-HT(1)
      HT(5)=0.0
      HT(6)=-2.*T
C
```

```
      PZT=HT(1)*ZZ(1)+HT(2)*ZZ(2)+HT(3)*ZZ(3)+HT(4)*ZZ(4)
      PZS=HS(1)*ZZ(1)+HS(2)*ZZ(2)+HS(3)*ZZ(3)+HS(4)*ZZ(4)
      PRS=HS(1)*RR(1)+HS(2)*RR(2)+HS(3)*RR(3)+HS(4)*RR(4)
      PRT=HT(1)*RR(1)+HT(2)*RR(2)+HT(3)*RR(3)+HT(4)*RR(4)
      XJ=PRS*PZT-PRT*PZS
C
      PSR=PZT/XJ
      PTR=-PZS/XJ
      PSZ=-PRT/XJ
      PTZ=PRS/XJ
C
      DO 100 I=1,6
      HR(I)=PSR*HS(I)+PTR*HT(I)
  100 HZ(I)=PSZ*HS(I)+PTZ*HT(I)
      R=H(1)*RR(1)+H(2)*RR(2)+H(3)*RR(3)+H(4)*RR(4)
      IF(NPAR(5).NE.0) R=THICK
C
C     FORM STRAIN DISPLACEMENT MATRIX
C
      DO 200 K=1,6
      I=II(K)
      J=JJ(K)
      B(1,I)=HR(K)
      B(2,J)=HZ(K)
C
C     TEST FOR HOOP STRAIN EVALUATION (AXISYMMETRIC SOLID)
C
      IF(NPAR(5).GT.0) GO TO 190
C        SET HOOP STRAIN .EQ. RADIAL STRAIN IF ON C/L AXIS
      IF(R.LT.1.0E-6)
     *B(3,I)=B(1,I)
C
      IF(R.GT.1.0E-6)
     *B(3,I)=H(K)/R
C
  190 CONTINUE
      B(4,I)=HZ(K)
  200 B(4,J)=HR(K)
C
      FAC=XJ*R
      RETURN
      END
      SUBROUTINE GMTN (FF,IFF,XM,MASS,NEQB,NFN,NBLOCK)
C
C     CALLED BY?  HISTRY
C
      COMMON / JUNK / NARB,NGM,JFN(3),JAT(3),IFILL1(422)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
      DIMENSION FF(NEQB,NFN),IFF(NEQB,NFN),MASS(NEQB),XM(NEQB)
C
C     GROUND MOTION EFFECTS
C
      IF(MODEX.EQ.1) GO TO 20
      JT=4
      IT=2
```

```
      REWIND IT
      REWIND JT
      REWIND 3
      REWIND 9
C
   20 CONTINUE
      READ (5,1000) JFN,JAT
      DO 100 I=1,3
      IF(JAT(I)) 50,50,100
   50 JAT(I)=1
  100 CONTINUE
      WRITE (6,2000) JFN,JAT
      IF(MODEX.EQ.1) RETURN
C
      NNN=NFN*NEQB
      DO 500 N=1,NBLOCK
C
      READ (3) MASS
      READ (9) XM
C
      IF(NARB.EQ.0) GO TO 200
      READ (IT) FF,IFF
      GO TO 300
  200 DO 250 I=1,NNN
      FF(I,1)=0.0D0
  250 IFF(I,1)=0
C
  300 DO 400 I=1,NEQB
      J=MASS(I)
      IF (J .LE. 0) GO TO 400
      JJ=JFN(J)
      IF(JJ.LE.0) GO TO 400
      FF(I,JJ)=-XM(I)
      IFF(I,JJ)=JAT(J)
  400 CONTINUE
C
  500 WRITE (JT) FF,IFF
C
      RETURN
C
 1000 FORMAT (6I5)
 2000 FORMAT (//// 26H GROUND ACCELERATION INPUT, // 28X,
     1 11HX-DIRECTION,2X,11HY-DIRECTION,2X,11HZ-DIRECTION, //
     2  26H TIME FUNCTION NUMBER(S) =, 3(10X,I3) /
     3  26H  ARRIVAL TIME NUMBER(S) =, 3(10X,I3) / 1X)
C
      END
      SUBROUTINE GROUND (FF,IFF,XM,MASS,NEQ,NFN)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE MODIFIES THE FUNCTION MULTIPLIERS AND ARRIVAL TIME
C     ARRAYS TO ACCOMODATE INPUT GROUND MOTION.
C
C     *XM*        / TAPE3 /   ADDMAS      /
```

```
C      *MASS*     / TAPE7    EMIDS      /
C      *FF*,*IFF* / TAPE2 /  PLOAD      /
C
       COMMON /JUNK/  JFN(3),JAT(3),IFILL1(424)
       COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
       DIMENSION FF(NEQ,NFN),IFF(NEQ,NFN),XM(NEQ),MASS(NEQ)
C
       IF(MODEX.EQ.1) GO TO 20
C
       NT=3
       IT=2
       KT=7
       REWIND NT
       REWIND KT
       REWIND IT
C
C      READ GROUND MOTION FUNCTION REFERENCES AND ARRIVAL TIMES
C
   20 READ (5,1000) JFN,JAT
       DO 100 I=1,3
       IF(JAT(I)) 50,50,100
   50 JAT(I)=1
  100 CONTINUE
       WRITE (6,2000) JFN,JAT
C
       IF(MODEX.EQ.1) RETURN
C
       READ (KT) MASS
       READ (NT) XM
       READ (IT) FF,IFF
       REWIND IT
C
C      MODIFY FUNCTION MULTIPLIERS AND ARRIVAL TIMES DUE TO
C      INPUT GROUND ACCELERATION(S)
C
       DO 400 I=1,NEQ
       J=MASS(I)
       IF(J.EQ.0)  GO TO 400
       JJ=JFN(J)
       IF(JJ.LE.0)  GO TO 400
       FF(I,JJ) =-XM(I)
       IFF(I,JJ)=JAT(J)
  400 CONTINUE
C
       WRITE (IT) FF,IFF
       RETURN
C
C      F O R M A T S
C
 1000 FORMAT (6I5)
 2000 FORMAT (38H1G R O U N D   M O T I O N   I N P U T, // 21X,
     1        9HDIRECTION, / 21X,1HX,3X,1HY,3X,1HZ, /
     2        19H FUNCTION NUMBERS =, I3,2I4 /
     3        19H ARRIVAL TIMES    =, I3,2I4 // 1X)
```

```
      END
      SUBROUTINE HISTRY
C
C     CALLS?  LOAD1,EMID,GMTN,LOAD2,RESPON,DISPLR,STRSD1
C     CALLED BY?  MAIN
C
C     TIME HISTORY RESPONSE CALCULATIONS
C
      COMMON /SOL/ NBLOCK,NEQB,LL,NF,LB,IFILL4(6)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
       COMMON / EM / AT(1058),IFILL2(3022)
      COMMON / DYN / NT,NOT,DAMP,DT,IFILL5(6)
      COMMON / JUNK / NARB,NGM,IFILL1(428)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
C
      REAL T(5),TT
C
      COMMON /one/ A(1)
c***      COMMON A(7100)
C
C
C***      CALL TTIME(T(1))
      READ (5,1000) NFN,NGM,NAT,NT,NOT,DT,DAMP
      IF(NAT.EQ.0) NAT=1
      WRITE (6,1010)
      WRITE (6,2000) NFN,NGM,NAT,NT,NOT,DT,DAMP
C
C     DYNAMIC LOADS
C
      N2=N1+6*NUMNP
      N3=N2+NFN*NEQB
      N4=N3+NFN*NEQB
      IF(N4.GT.MTOT) CALL ERROR(MTOT-N4)
      CALL LOAD1 (A(N1),A(N2),A(N3),NUMNP,NEQB,NFN)
      IF(NGM.EQ.0) GO TO 300
C
C     ADD GROUND MOTION EFFECTS
C
      IF(MODEX.LT.1)
     *CALL EMID (A(N1),A(N2),NUMNP,NEQB)
      N2=N1+NEQB*NFN
      N3=N2+NEQB*NFN
      N4=N3+NEQB
      N5=N4+NEQB
      IF(N5.GT.MTOT) CALL ERROR(N5-MTOT)
C
      CALL GMTN (A(N1),A(N2),A(N3),A(N4),NEQB,NFN,NBLOCK)
C
  300 N2=N1+NFN*NF*NAT
      N3=N2+NEQB*NF
      N4=N3+NEQB*NFN
      N5=N4+NEQB*NFN
      IF(N5.GT.MTOT) CALL ERROR (N5-MTOT)
C
      N6=N2+NT*NFN
```

```
      MAX=(MTOT-N6)/2
      N7=N6+MAX
C
      N8=N6+NT
       IF(N8.GT.MTOT) CALL ERROR (N8-MTOT)
      CALL LOAD2 (A(N2),A(N3),A(N4),A(N2),A(N6),A(N7),
     .            A(N6),NEQB,NF,NFN,NT,MAX,NBLOCK,NAT)
C
C     NORMAL RESPONSE
C
C***       CALL TTIME(T(2))
      NDS=(NT-1)/NOT
      N2=N1+NF
      N3=N2+NT
      N4=N3+NF*NDS
      IF(N4.GT.MTOT) CALL ERROR(N4-MTOT)
      IF(MODEX.EQ.1) GO TO 320
      CALL RESPON (A(N1),A(N2),A(N3),NF,NT,NDS)
C
C     DISPLACEMENT RESPONSE
C
C***  320 CALL TTIME(T(3))   !320 IS TRANSFERED TO THE NEXT LINE
320      NSB=NEQB*NBLOCK
      N2=N1+8*NF
      N3=N2+NF*NDS
      IF(N3.GT.MTOT) CALL ERROR(N3-MTOT)
      CALL DISPLR (A(N1),A(N1),A(N2),A(N2),NEQB,NF,NDS,NUMNP,NBLOCK,NSB)
C
C     STRESS RESPONSE
C
C***       CALL TTIME(T(4))
C
      N2=N1+NELTYP
      N3=N2+8*NF
      N4=N3+NSB*NF
      N5=N3+NF*NDS
      IF(N4.GT.MTOT) CALL ERROR(N4-MTOT)
      IF(N5.GT.MTOT) CALL ERROR(N5-MTOT)
      CALL STRSD1(A(N1),A(N2),A(N3),A(N3),NF,NSB,NDS,NEQB,NBLOCK)
C***       CALL TTIME(T(5))
C
C     COMPUTE AND PRINT THE SOLUTION TIME LOG
C
      TT=0.
      DO 100 I=1,4
      T(I)=T(I+1)-T(I)
  100 TT=TT + T(I)
      T(5) = TT
      WRITE (6,3000) T
C
      RETURN
C
 1000 FORMAT (5I5,2F10.0)
 1010 FORMAT (1H1,//49H F O R C E D   R E S P O N S E   A N A L Y S I S
     1//)
```

```
 2000 FORMAT (20HOCONTROL INFORMATION  ,//
     2  25H NUMBER OF TIME FUNCTIONS,  2X,1H=,I5 /
     3  24H GROUND MOTION INDICATOR,   3X,1H=,I5 /
     4  14H   EQ.0,  NONE, /
     5  22H   EQ.1,  GROUND INPUT, /
     6  24H NUMBER OF ARRIVAL TIMES,   3X,1H=,I5 /
     7  21H NUMBER OF TIME STEPS,      6X,1H=,I5 /
     8  22H OUTPUT PRINT INTERVAL,     5X,1H=,I5 /
     9  10H TIME STEP,                17X,1H=,F11.5 /
     A  15H DAMPING FACTOR,          12X,1H=,F11.5   )
 3000 FORMAT (48H1F O R C E D   R E S P O N S E   T I M E   L O G,///
     . 33H  FORM DYNAMIC LOADS............  ,F8.2 //
     . 33H  MODAL RESPONSE................  ,F8.2 //
     . 33H  DISPLACEMENT OUTPUT...........  ,F8.2 //
     . 33H  STRESS OUTPUT.................  ,F8.2 //
     . 33H  TOTAL FOR RESPONSE ANALYSIS...  ,F8.2 //)
C
      END
      SUBROUTINE INDLY (FF,IFF,AT,NEQ,NFN,NAT,MAXD)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE READS *NAT* ARRIVAL TIME VALUES FROM DATA INPUT.
C     ARRIVAL TIMES ARE CONVERTED TO THE NEAREST (INTEGER) TIME STEP
C     NUMBER, AND ARRIVAL TIME REFERENCES (PREVIOUSLY STORED IN
C     *IFF*) ARE REPLACED BY THE TIME STEP NUMBERS.
C
      COMMON /DYN/   NT,NOT,ALFA,DT,BETA,IFILL1(4)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
      DIMENSION      FF(NEQ,NFN),IFF(NEQ,NFN),AT(NAT)
C
      IF(MODEX.EQ.1) GO TO 50
C
      KT=2
      REWIND KT
C
C     READ ARRIVAL TIME DATA
C
   50 READ  (5,1002)  (  AT(I),I=1,NAT)
      WRITE (6,2004)  (I,AT(I),I=1,NAT)
      MAXD=0
C
      IF(MODEX.EQ.1) RETURN
C
      DO 100 I=1,NAT
  100 AT(I)=AT(I)/DT
C
      READ (KT)  FF,IFF
      REWIND KT
C
      DO 300 NF=1,NFN
      DO 200 N=1,NEQ
      J=IFF(N,NF)
      JAT=AT(J)
```

```
      IF((AT(J)-JAT).GE.0.5)  JAT=JAT+1
      JAT=JAT+1
      IF(JAT.GT.MAXD)  MAXD=JAT
C
C     NOTE    *MAXD* IS THE LARGEST TIME STEP NUMBER ASSOCIATED WITH
C             ANY ONE OF THE INPUT DELAY TIMES.  *MAXD* IS USED FOR
C             CORE STORAGE ALLOCATION DURING LOAD VECTOR CALCULATIONS.
C
  200 IFF(N,NF) = JAT
  300 CONTINUE
C
      WRITE (KT) FF,IFF
      RETURN
C
C     F O R M A T S
C
 1002 FORMAT (8F10.2)
 2004 FORMAT (//// 38H A R R I V A L   T I M E   V A L U E S, //
     1        6H INPUT,5X,12HARRIVAL TIME,/ 6H ORDER,12X,5HVALUE, //
     2        (I6,E17.4) )
C
      END
      SUBROUTINE INOUT (IDIS,ID,ISTR,NUMNP)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE PROCESSES OUTPUT REQUESTS FOR DISPLACEMENTS AND
C     ELEMENT STRESS COMPONENTS.  TAPE9 IS USED TO SAVE OUTPUT SET
C     REQUESTS (8 REQUESTS PER SET), AND TAPE8 IS USED TO SAVE THE
C     STRESS-DISPLACEMENT TRANSFORMATIONS FOR ELEMENT STRESSES WHICH
C     ARE REQUESTED FOR OUTPUT.
C
C     NOTE   *ID* AND *ISTR* ARE EQUIVALENCED IN BLANK COMMON BY THE
C            CALLING PROGRAM
C
      DIMENSION      IDIS(1),ID(NUMNP,6),ISTR(1),KLM(8,63),SSA(8,63)
C
      COMMON /JUNK/  KK1,KK2,ISP1,ISP2,NSD,NSS,IC(6),KD(2,8),IS(12),
     1              IDUM(32),NUM(100),IFILL1(258)
      COMMON /ELPAR/ NPAR(14),IDUM1,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,
     1              NEQ
      COMMON /EM/ SA(42,63),ND,NS,LM(63)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
C
      IF(MODEX.EQ.1) GO TO 10
C
      REWIND 1
      REWIND 8
      REWIND 9
      GO TO 20
C
C     RESTORE MASTER INDEX ARRAY *ID*
C
   10 REWIND 1
      REWIND 2
```

```
      READ  (2) ID
      GO TO 25
   20 CONTINUE
      READ (8) ID
      REWIND 8
C
   25 L=0
      K=0
C
C     PROCESS DISPLACEMENT REQUESTS
C
      WRITE (6,1005)
C
C        1. OUTPUT TYPE
C
      READ  (5,2000) KK1,ISP1
      WRITE (6,4000) KK1,ISP1
      WRITE (6,1006)
C
C        2. CARD READING LOOP (TERMINATE READING IF ZERO NODE IS READ)
C
  100 READ  (5,2000) NP,IC
      WRITE (6,2001) NP,IC
      IF(NP.GT.0) GO TO 110
      IF(L.EQ.0) GO TO 200
C
C        3. SAVE LAST OUTPUT SET
C
      IF(MODEX.EQ.0)
     *WRITE (9) KD,L
      GO TO 200
C
C        4. CONSIDER SIX (6) POSSIBLE REQUESTS ON THIS CARD
C
  110 IF(NP.LE.NUMNP) GO TO 112
      WRITE (6,3010) NP
 3010 FORMAT (19H0*** ERROR   NODE (,I5,15H) IS TOO LARGE., / 1X)
      STOP
  112 DO 150 I=1,6
      II=IC(I)
      IF(II.EQ.0 .OR. II.GT.6) GO TO 100
      K=K+1
      L=L+1
C
C        5. SAVE NODE NUMBER AND COMPONENTS NUMBER IN *KD*
C
      KD(1,L)=NP
      KD(2,L)=II
      JJ=ID(NP,II)
      IF(JJ.GT.0) GO TO 130
      L = L-1
      K=K-1
      GO TO 140
  130 IDIS(K)=JJ
  140 IF(L.LT.8) GO TO 150
```

```
C
C        6. SAVE THIS OUTPUT SET CONSISTING OF 8 REQUESTS
C
      IF(MODEX.EQ.0)
     *WRITE (9) KD,L
      L=0
  150 CONTINUE
      GO TO 100
C
C        7. SAVE THE TOTAL NUMBER OF DISPLACEMENT COMPONENTS REQUESTED
C           FOR OUTPUT
C
  200 NSD=K
C
C     PROCESS ELEMENT STRESS COMPONENT REQUESTS
C
      WRITE (6,3000)
C
C        1. OUTPUT TYPE
C
      READ  (5,2000) KK2,ISP2
      WRITE (6,4000) KK2,ISP2
      K = 1
      ISTR(1) = 0
C
C        2. CONSIDER EACH ELEMENT TYPE
C
      DO 500 N=1,NELTYP
C
      READ (1) NPAR
      IF(MODEX.EQ.0)
     *WRITE (9) NPAR
C
C        3. LABEL ELEMENT TYPE
C
      WRITE (6,3001) NPAR(1)
C
C        4. READ FIRST ELEMENT REQUEST IN THIS GROUP
C
      READ  (5,2000) NEL,IS
      WRITE (6,2001) NEL,IS
      NUME=NPAR(2)
      L=0
      NUM(N)=0
C
C        5. LOOP ON THE TOTAL NUMBER OF ELEMENTS OF THIS (THE N-TH)
C           TYPE.  COMPACT STRESS TRANSFORMATIONS WHEN ELEMENT NUMBER
C           MATCH IS FOUND.  ELEMENT OUTPUT REQUESTS ARE EXPECTED IN
C           ASCENDING ELEMENT NUMBER ORDER.  ANY REQUESTED ELEMENT
C           NUMBER LESS THAN PREVIOUSLY READ NUMBER WILL FORCE THIS
C           LOOP TO BE EXHAUSTED (I.E., TERMINATE WITH ZERO ELEMENT).
C
      DO 400 M=1,NUME
C
      IF(MODEX.EQ.0)
```

```
     *READ (1) ND,NS,(LM(I),I=1,ND),((SA(I,J),I=1,NS),J=1,ND)
      IF(MODEX.EQ.1)
     *READ (1) ND,NS,(LM(I),I=1,ND)
C
      IF(NEL.NE.M)  GO TO 400
      KS = NS
      IF(KS.GT.12) KS = 12
C
C        6. CONSIDER 12 (MAXIMUM) REQUESTS FOR THIS ELEMENT
C
      DO 300 I=1,KS
      II=IS(I)
      IF(II.EQ.0)  GO TO 350
      IF(II.GT.NS) GO TO 300
      L=L+1
C
C        7. SAVE THE ELEMENT NUMBER AND STRESS COMPONENT NUMBER IN *KD*
C
      KD(1,L)=NEL
      KD(2,L)=II
C
C        8. SAVE STRESS TRANSFORMATION FOR COMPONENT *II* IN *SSA* AND
C           COMPUTE (AND SAVE IN *KLM*) THE LOCATION IN VECTOR *ISRT*
C           WHICH CONTAINS THE EQUATION NUMBER FOR THE J-TH ELEMENT
C           DEGREE OF FREEDOM
C
      DO 250 J=1,ND
      IF(MODEX.EQ.0)
     *SSA(L,J) = SA(II,J)
      KLM(L,J)=0
      JJ=LM(J)
      IF(JJ.LE.0) GO TO 250
C
C        9. CHECK FOR EQUATION NUMBER *JJ* IN ISTR*.  IF FOUND, SET
C           *KLM* TO LOCATION WHERE FOUND.  IF NOT FOUND, EXTEND *ISTR*
C           TO ACCOMODATE THE NEW EQUATION NUMBER.
C
      DO 220 NK=1,K
      IF(ISTR(NK).NE.JJ)  GO TO 220
      KLM(L,J)=NK
      GO TO 250
  220 CONTINUE
      ISTR(K)=JJ
      KLM(L,J)=K
      K=K+1
      ISTR(K)=0
  250 CONTINUE
C
C        10. SAVE OUTPUT REQUESTS AND TRANSFORMATIONS TO ALLOW STRESS
C            RECOVERY ONCE DISPLACEMENTS ARE KNOWN
C
      IF(L.LT.8)  GO TO 300
      IF(MODEX.EQ.1) GO TO 290
      WRITE (9) KD,L
      WRITE (8) ND,((SSA(II,JJ),II=1,8),JJ=1,ND),
```

```
     1              ((KLM(II,JJ),II=1,8),JJ=1,ND)
  290 L=0
      NUM(N)=NUM(N)+1
  300 CONTINUE
C
C        11. READ NEXT REQUEST AND BRANCH BACK TO SCAN FOR NEW MATCH
C
  350 READ  (5,2000) NEL,IS
      WRITE (6,2001) NEL,IS
C
  400 CONTINUE
C
C        12. SAVE FINAL STRESS OUTPUT RECORD
C
      IF(L.EQ.0)  GO TO 500
      IF(MODEX.EQ.1) GO TO 490
      WRITE (9) KD,L
      WRITE (8) ND,((SSA(II,JJ),II=1,8),JJ=1,ND),
     1              ((KLM(II,JJ),II=1,8),JJ=1,ND)
  490 NUM(N) = NUM(N) + 1
  500 CONTINUE
C
C        13. SAVE THE TOTAL NUMBER OF DISPLACEMENTS (I.E., ENTRIES IN
C            *ISTR*) REQUIRED TO RECOVER ELEMENT STRESSES.
C
      NSS=K-1
C
C     SHIFT *ISTR* BACK IN BLANK COMMON ADJACENT TO *IDIS(NSD)* SO
C     THAT *IDIS* AND ISTR* ARE CONTIGUOUS IN STORAGE.
C
      IF(NSS.LT.1) RETURN
      DO 550 L=1,NSS
      J = NSD+L
  550 IDIS(J) = ISTR(L)
C
      RETURN
C
C     F O R M A T S
C
 1005 FORMAT (44HID I S P L A C E M E N T   C O M P O N E N T,3X,
     1        29HO U T P U T   R E Q U E S T S, // 1X)
 1006 FORMAT (4X,4HNODE,2X,22HDISPLACEMENT COMPONENT, / 2X,6HNUMBER,
     1        6(3X,1H*), / 1X)
 2000 FORMAT (13I5)
 2001 FORMAT (I8,12I4)
 3000 FORMAT (46HIS T R E S S   C O M P O N E N T   O U T P U T,3X,
     1        15HR E Q U E S T S, // 1X)
 3001 FORMAT (// 6X,23HE L E M E N T   T Y P E,3X,1H(,I2,1H), //
     1        8H ELEMENT,9X,33HDESIRED ELEMENT STRESS COMPONENTS, /
     2        8H  NUMBER,12(3X,1H*), / 1X)
 4000 FORMAT (// 25H CODE FOR OUTPUT TYPE   =, I2 /
     1            3X,19HEQ.1, HISTORY TABLE,      /
     2            3X,18HEQ.2, PRINTER PLOT,       /
     3            3X,17HEQ.3, MAXIMA ONLY,        /
     4            25H PRINTER PLOT SPACING   =, I2 / 1X)
```

```
C
      END
      SUBROUTINE INP21 (NUMMAT,MAXTP,NORTHO,NDLS,NOPSET,NT8SV,NUMNP,X,
     1          Y,Z,DEN,RHO,NTP,EE,DCA,NFACE,LT,PWA,LOC,MAXPTS)
C
C     CALLED BY ? THDFE
C     CALLS ? VECTR2,CROSS2
C
C
C
C     THIS ROUTINE READS AND PRINTS ALL 21-NODE SOLID ELEMENT DATA
C     BETWEEN THE CONTROL CARD AND THE ELEMENT DATA CARDS
C
C
      COMMON / JUNK/  XLF(4),YLF(4),ZLF(4),TLF(4),PLF(4),FILL1(22),V2(3)
      COMMON /EXTRA/ MODEX,NT8
C
      DIMENSION   X(1),Y(1),Z(1),DEN(1),RHO(1),NTP(1),EE(MAXTP,13,1),
     1            DCA(3,3,1),NFACE(1),LT(1),PWA(7,1),LOC(7,1),
     2            MAXPTS(1)
      DIMENSION   HED(6)
C
C     READ AND PRINT OF MATERIAL PROPERTIES
C
      WRITE (6,3000)
C
      DO 10 I=1,NUMMAT
C
      READ (5,1001) M,NTP(I),DEN(I),RHO(I),(HED(N),N=1,6)
C
C     SET DEFAULT VALUES IF REQUIRED AND CHECK FOR INPUT ERRORS
C
      IF(RHO(I).EQ.0.0) RHO(I) = DEN(I) / 386.4
      IF(NTP(I).EQ.0) NTP(I) = 1
C
      WRITE (6,3002) M,NTP(I),DEN(I),RHO(I),(HED(N),N=1,6)
C
      IF(I.EQ.M) GO TO 2
      WRITE (6,4001)
      STOP
C
    2 IF(NTP(M).LE.MAXTP) GO TO 4
      WRITE (6,4002) MAXTP
      STOP
    4 NT = NTP(M)
C
C     READ PROPERTIES FOR EACH TEMPERATURE
C
      DO 6 K=1,NT
      READ (5,1002) (EE(K,L,M),L=1,13)
      WRITE (6,3003) (EE(K,L,M),L=1,13)
    6 CONTINUE
C
C     TEMPERATURE CARDS MUST BE ASCENDING ORDER
C
```

```
      IF(NT.EQ.1) GO TO 10
      DO 8 J=2,NT
      IF(EE(J,1,M).GT.EE(J-1,1,M)) GO TO 8
      WRITE (6,4003)
      STOP
    8  CONTINUE
   10  CONTINUE
C***  DATA PORTHOLE SAVE
      IF(NT8SV.EQ.0) GO TO 12
      DO 11 M=1,NUMMAT
      WRITE (NT8) M,NTP(M),DEN(M),RHO(M)
      NT = NTP(M)
      WRITE (NT8) ((EE(K,L,M),L=1,13),K=1,NT)
   11 CONTINUE
C***
C
C     MATERIAL AXIS ORIENTATION SETS
C
   12 IF(NORTHO.EQ.0) GO TO 21
C
      WRITE (6,3004)
C
      DO 20 M=1,NORTHO
      READ (5,1003) N,NI,NJ,NK
      WRITE (6,3005) N,NI,NJ,NK
C
C***  DATA PORTHOLE SAVE
      IF(NT8SV.EQ.1)
     *WRITE (NT8)    N,NI,NJ,NK
C***
C     CHECK FOR ADMISSABILITY OF DATA
C
      IF(N.EQ.M) GO TO 13
      WRITE (6,4004)
      STOP
C
   13 IF(NI.GT.0 .AND. NI.LE.NUMNP) GO TO 5015
      L = NI
 5014 WRITE (6,4005) L
      STOP
 5015 IF(NJ.GT.0 .AND. NJ.LE.NUMNP) GO TO 5016
      L = NJ
      GO TO 5014
 5016 IF(NK.GT.0 .AND. NK.LE.NUMNP) GO TO 14
      L = NK
      GO TO 5014
   14 CONTINUE
C
C     GENERATE DIRECTION COSINE ARRAY FOR THIS DATA SET
C
      CALL VECTR2 (DCA(1,1,M),X(NI),Y(NI),Z(NI),X(NJ),Y(NJ),Z(NJ),IERR)
      IF(IERR.EQ.0) GO TO 16
      WRITE (6,4006)
      STOP
   16 CALL VECTR2 (V2,X(NI),Y(NI),Z(NI),X(NK),Y(NK),Z(NK),IERR)
```

```
      IF(IERR.EQ.0) GO TO 17
      WRITE (6,4007)
      STOP
   17 CALL CROSS2 (DCA(1,1,M),V2,DCA(1,3,M),IERR)
      IF(IERR.EQ.0) GO TO 18
      WRITE (6,4008)
      STOP
   18 CALL CROSS2 (DCA(1,3,M),DCA(1,1,M),DCA(1,2,M),IERR)
      IF(IERR.EQ.0) GO TO 20
      WRITE (6,4009)
      STOP
   20 CONTINUE
C
C     READ AND PRINT DISTRIBUTED SURFACE LOAD DATA
C
   21 IF(NDLS.EQ.0) GO TO 31
C
      WRITE (6,3006)
C
      DO 30 M=1,NDLS
C
      READ  (5,1004) N,NFACE(M),LT(M)
      WRITE (6,3007) N,NFACE(M),LT(M)
C
C     CHECK FOR DATA ADMISSABILITY
C
      IF(N.EQ.M) GO TO 22
      WRITE (6,4010)
      STOP
   22 IF(NFACE(M).GE.1 .AND. NFACE(M).LE.6) GO TO 23
      WRITE (6,4011)
      STOP
   23 IF(LT(M).EQ.0) LT(M) = 1
      IF(LT(M).GE.1 .AND. LT(M).LE.2) GO TO 24
      WRITE (6,4012)
      STOP
   24 IF(LT(M).EQ.2) GO TO 26
      READ (5,1005) (PWA(I,M),I=1,4)
      DO 25 I=2,4
   25 IF(PWA(I,M).EQ.0.0) PWA(I,M) = PWA(1,M)
      WRITE (6,3008) (PWA(I,M),I=1,4)
      GO TO 30
   26 READ (5,1005) (PWA(I,M),I=1,7)
      WRITE (6,3009) (PWA(I,M),I=1,7)
   30 CONTINUE
C
C***  DATA PORTHOLE SAVE
      IF(NT8SV.EQ.0) GO TO 5031
      DO 5030 M=1,NDLS
      WRITE (NT8) NFACE(M),LT(M),(PWA(I,M),I=1,7)
 5030 CONTINUE
 5031 CONTINUE
C***
C
C     READ AND PRINT OF STRESS OUTPUT REQUEST LOCATION SETS
```

```
C
   31 IF(NOPSET.EQ.0) GO TO 49
C
      WRITE (6,3010) (I,I=1,7)
C
      DO 40 M=1,NOPSET
      READ (5,1006)  (LOC(I,M),I=1,7)
      WRITE (6,3011) M,(LOC(I,M),I=1,7)
C
      L = 0
      DO 35 J=1,7
      IF(LOC(J,M).EQ.0) GO TO 36
      L = L + 1
      IF(LOC(J,M).GE.1 .AND. LOC(J,M).LE.27) GO TO 35
      WRITE (6,4013) J
      MODEX = 1
      GO TO 36
   35 CONTINUE
C
   36 IF(L.GT.0) GO TO 37
      L = 1
      LOC(1,M) = 21
   37 MAXPTS(M) = L
C
   40 CONTINUE
C***  DATA PORTHOLE SAVE
      IF(NT8SV.EQ.1)
     *WRITE (NT8) ((LOC(I,J),I=1,7),J=1,NOPSET)
C***
C
C     ELEMENT LOAD CASE MULTIPLIERS
C
   49 WRITE (6,3012)
C
      READ (5,1007) XLF,YLF,ZLF,TLF,PLF
      WRITE (6,3013) XLF,YLF,ZLF,TLF,PLF
C***  DATA PORTHOLE SAVE
      IF(NT8SV.EQ.1)
     *WRITE (NT8)   XLF,YLF,ZLF,TLF,PLF
C***
C
      RETURN
C
C     FORMATS
C
 1001 FORMAT (2I5,2F10.0,6A6)
 1002 FORMAT (7F10.0/6F10.0)
 1003 FORMAT (4I5)
 1004 FORMAT(3I5)
 1005 FORMAT (7F10.0)
 1006 FORMAT (7I5)
 1007 FORMAT (4F10.0)
C
 3000 FORMAT (//38H MATERIAL PROPERTY TABLES                    )
 3002 FORMAT (//22HOMATERIAL NUMBER   = (,I3,1H),/
```

```
     1          10H NUMBER OF,                          /
     2          23H TEMPERATURE POINTS = (,I3,1H),/
     3          23H WEIGHT DENSITY     = (    ,E12.4,1H),/
     4          23H MASS  /DENSITY     = (,E12.4,1H),/
     5          23H IDENTIFICATION     = (,6A6,1H),//
     6 1X,11HTEMPERATURE,9X,3HE11,9X,3HE22,9X,3HE33,4X,3HV12,4X,3HV13,
     7 4X,3HV23,8X,3HG12,8X,3HG13,8X,3HG23,3X,7HALPHA-1,3X,7HALPHA-2,
     8 3X,7HALPHA-3,/1X)
 3003 FORMAT (F12.2,3F12.1,3F7.3,3F11.1,3E10.3)
 3004 FORMAT (//50H M A T E R I A L  A X I S  O R I E N T A T I O N    ,
     1 3X,9HT A B L E    ,//
     2 28H    SET   NODE   NODE   NODE   ,/
     3 28H NUMBER     NI     NJ     NK, / 1X)
 3005 FORMAT (4I7)
 3006 FORMAT(//51H D I S T R I B U T E D  S U R F A C E  L O A D      ,
     1          11HT A B L E    ,/,1X )
 3007 FORMAT (//7X,27HLOAD SET NUMBER            = ,I6 /
     1          7X,27HLOAD SURFACE ELEMENT FACE = ,I6 /
     1          7X,27HLOAD TYPE CODE            = ,I6/1X)
 3008 FORMAT (12H DISTRIBUTED,  11X,4HP(1),11X,4HP(2),11X,4HP(3),11X,
     1          4HP(4), / 4X,8HPRESSURE,4F15.3)
 3009 FORMAT (12H HYDROSTATIC,10X,5HGAMMA,11X,4HX(S),11X,4HY(S),11X,
     1          4HZ(S),11X,4HX(N),11X,4HY(N),11X,4HZ(N), /
     2          4X,8HPRESSURE, 7F15.3)
 3010 FORMAT (//51H S T R E S S  O U T P U T  R E Q U E S T  T A B L E ,
     *  //
     *8H    SET ,7(2X,5HPOINT), / 8H NUMBER ,7(4X,1H(,I1,1H)),/ 1X)
 3011 FORMAT (I8,7I7)
 3012 FORMAT (///34H E L E M E N T  L O A D  C A S E         ,3X,
     1 21HM U L T I P L I E R S        ,//
     *                    31X,6HCASE A,4X,6HCASE B,4X,6HCASE C,
     2 4X,6HCASE D,/1X)
 3013 FORMAT (
     1 27H X-DIRECTION GRAVITY =      ,4F10.2/
     2 27H Y-DIRECTION GRAVITY =      ,4F10.2/
     3 27H Z-DIRECTION GRAVITY =      ,4F10.2/
     4 27H THERMAL LOADING     =      ,4F10.2/
     5 27H PRESSURE LOADING    =      ,4F10.2  //1X)
C
 4001 FORMAT (40HOERROR***   MATERIAL CARDS OUT OF ORDER.,/1X)
 4002 FORMAT (52HOERROR***   NUMBER OF TEMPERATURE CARDS EXCEEDS USER,
     1 10H MAXIMUM (,I4,2H)., / 1X)
 4003 FORMAT (51HOERROR***   TEMPERATURES MUST BE INPUT IN ASCENDING  ,
     1 7H ORDER., / 1X)
 4004 FORMAT (47HOERROR***   AXIS ORIENTATION CARD OUT OF ORDER.,/1X)
 4005 FORMAT (36HOERROR***   UNDEFINED NODE NUMBER = ,I5 / 1X)
 4006 FORMAT (38HOERROR***   VECTOR IJ HAS ZERO LENGTH.,/1X)
 4007 FORMAT (38HOERROR***   VECTOR IK HAS ZERO LENGTH.,/1X)
 4008 FORMAT (43HOERROR***   IJ AND IK VECTORS ARE PARALLEL.,/1X)
 4009 FORMAT (43HOERROR***   F3 AND F1 VECTORS ARE PARALLEL.,/1X)
 4010 FORMAT (50HOERROR***   SET NUMBERS MUST BE IN ASCENDING ORDER,/1X)
 4011 FORMAT (40HOERROR***   INVALID SURFACE FACE NUMBER.,/1X)
 4012 FORMAT (30HOERROR***   INVALID LOAD TYPE.,/1X)
 4013 FORMAT (42HOERROR***   INVALID OUTPUT POINT NUMBER = , I6 / 1X)
C
```

```
C
      END
      SUBROUTINE INTHIS (NLP,P,NFN,MXLP,KN)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE READS TIME FUNCTIONS FROM CARD INPUT.
C
C     NUMBER OF TIME POINTS PER FUNCTION STORED IN *NLP*, AND *P*
C     CONTAINS  (T,F(T))  PAIRS FOR EACH FUNCTION.  *MXLP* IS THE
C     MAXIMUM NUMBER OF TABLE POINTS (ENTRIES) USED TO DESCRIBE ANY
C     ONE OF THE FUNCTIONS.
C
      DIMENSION      NLP(NFN),P(KN,1)
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /JUNK/  HED(12),IFILL1(406)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
      MXLP=0
      NF=1
C
C     CARD READING LOOP
C
      WRITE (6,2002)
   50 NF2=2*NF
      NF1=NF2-1
      READ (5,1000) NLP(NF),SFTR,HED
      IF (ABS(SFTR) .LT. 1.0D-8) SFTR=1.0D0
      IF(NLP(NF).GT.MXLP) MXLP = NLP(NF)
      WRITE (6,2000) NF,NLP(NF),SFTR,HED
      N3 = N2+KN*MXLP
      IF(N3.GT.MTOT) CALL ERROR (N3-MTOT)
C
      NN=NLP(NF)
      READ  (5,1001) (  P(NF1,L),P(NF2,L),L=1,NN)
      WRITE (6,2001) (L,P(NF1,L),P(NF2,L),L=1,NN)
C
      IF(MODEX.EQ.1) GO TO 105
C
C     SCALE FUNCTION VALUES
C
      DO 100 K=1,NN
  100 P(NF2,K) = P(NF2,K)* SFTR
C
C     TEST THE TIME VALUE FOR THE FIRST INPUT TABLE POINT.  THIS FIRST
C     POINT MUST BE AT TIME ZERO.
C
  105 IF (ABS(P(NF1,1)) .LT. 1.0D-8) GO TO 110
      WRITE (6,3000) NF
      STOP
  110 CONTINUE
      NF=NF+1
      IF(NF.LE.NFN)  GO TO 50
      RETURN
```

```
C
C     F O R M A T S
C
 1000 FORMAT (I5,F10.0,12A5)
 1001 FORMAT (12F6.0)
 2000 FORMAT (// 26H  TIME FUNCTION NUMBER = (,I3,1H), //
     1        5X,21HNUMBER OF POINTS   = (, I3,    1H), /
     2        5X,21HSCALE FACTOR       = (,E12.4,  1H), /
     3        5X,21HDESCRIPTION        = (, 12A5,  1H), //
     4        8X,5HINPUT,8X,4HTIME,4X,8HFUNCTION, / 8X,5HORDER,
     5        2(7X,5HVALUE), / 1X)
 2001 FORMAT (8X,I5,2E12.4)
 2002 FORMAT (36H1T I M E   F U N C T I O N   D A T A, / 1X)
 3000 FORMAT (30HO*** ERROR   FUNCTION NUMBER (,I4,10H) DOES NOT,
     1        20H BEGIN AT TIME ZERO., / 1X)
C
      END
       SUBROUTINE INVECT (VA,XM,IEQ,NBLOCK,NEQB,NV,IFPR)
C
C     CALLED BY?  SSPCEB
C
      COMMON /SOL/ IDUM(10),NFO
       COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
       DIMENSION VA(NEQB,NV),XM(NEQB),IEQ(1)
C
      NV1=NV-1-NFO
       KK=1
       IND=0
 90    NBV=KK*((NV1-1)/NBLOCK+1)
       IF  (NBV.GT.NEQB) NBV=NEQB
       IF  (NBV.EQ.NEQB)  IND=1
       NBVN=0
       ICOUNT=0
       LL=0
C
       REWIND NMASS
       REWIND NSTIF
 60    READ (NMASS) XM
      READ (NSTIF)  (VA(I,1),I=1,NEQB)
       ICOUNT=ICOUNT+1
       DO 20 I=1,NEQB
      IF  (VA(I,1)  .EQ. 0.0D0) GO TO 20
      VA(I,1)=XM(I)/VA(I,1)
 20    CONTINUE
C
       NNV=NEQB/NBV
       DO 40 L=1,NBV
       RT=0.0
       NN=L*NNV
       DO 34 I=1,NN
      IF  (VA(I,1)  .LT. RT) GO TO 34
      RT=VA(I,1)
       IJ=I
 34    CONTINUE
       DO 30 I=NN,NEQB
```

```
      IF (VA(I,1) .LE. RT) GO TO 30
      RT=VA(I,1)
       IJ=I
 30    CONTINUE
      IF (VA(IJ,1) .NE. 0.0D0) GO TO 32
       NBVN=NBVN+1
       GO TO 40
 32    LL=LL+1
       IEQ(LL)=(ICOUNT-1)*NEQB+IJ
       IF (LL.GE.NV1) GO TO 50
      VA(IJ,1)=0.0D0
 40    CONTINUE
       IF (IND.EQ.1) GO TO 45
       IF ((NBVN.EQ.0).OR.(ICOUNT.EQ.NBLOCK)) GO TO 45
       NBV=KK*((NV1-LL-1)/(NBLOCK-ICOUNT)+1)
       IF (NBV.GT.NEQB) NBV=NEQB
       NBVN=0
C
 45    IF (ICOUNT.LT.NBLOCK) GO TO 60
       IF (IND.EQ.1) GO TO 47
       KK=2*KK
       GO TO 90
   47 WRITE (6,1000)
       STOP
C
 50    REWIND NMASS
       REWIND NR
      REWIND NT
      NSH1=NFO+1
      NSH2=NFO+2
       DO 100 L=1,NBLOCK
       READ (NMASS) XM
      IF(NFO.LE.0) GO TO 115
      READ(NT) VA
  115 DO 120 I=1,NEQB
      VA(I,NSH1)=XM(I)
      DO 120 J=NSH2,NV
 120   VA(I,J)=0.0
      DO 140 K=1,NV1
      II=IEQ(K)
       NLE=(L-1)*NEQB
       NRI=L*NEQB
       IF (II-NLE) 140,140,160
 160   IF (NRI-II) 140,180,180
 180   II=II-NLE
      VA(II,K+NSH1)=1.
 140   CONTINUE
       WRITE (NR) VA
 100   CONTINUE
       IF (IFPR.EQ.1)
     * WRITE (6,1010)
       IF (IFPR.EQ.1)
     * WRITE (6,1020) (IEQ(I),I=1,NV1)
C
       RETURN
```

```
C
 1000 FORMAT (37HO***ERROR   SOLUTION STOP IN *INVECT*, / 12X,
     1        42HINSUFFICIENT NUMBER OF FINITE EIGENVALUES., / 1X)
 1010  FORMAT (20HOPRINT OF VECTOR IEQ  )
 1020  FORMAT (1HO,2016)
       END
      SUBROUTINE JACOBI (A,B,X,EIGV,D,N,RTOL,IFPR)
C
C     CALLED BY?  EIGSOL
C
       DIMENSION A(N,N),B(N,N),X(N,N),EIGV(N),D(N)
C
       NSMAX=15
       DO 10 I=1,N
       D(I)=A(I,I)/B(I,I)
 10    EIGV(I)=D(I)
       DO 30 I=1,N
       DO 20 J=1,N
 20    X(I,J)=0.
 30    X(I,I)=1.0
      IF(N.EQ.1) RETURN
       NSWEEP=0
       NR=N-1
C
C     WE START ITERATION
 40    NSWEEP=NSWEEP+1
      IF(IFPR.EQ.1)
     *WRITE (6,1000) NSWEEP
       EPS=(0.01**NSWEEP)**2
       DO 50 J=1,NR
       JJ=J+1
       DO 50 K=JJ,N
       TT=A(J,K)*A(J,K)
       TB=A(J,J)*A(K,K)
      EPTOLA=ABS(TT/TB)
       TT=B(J,K)*B(J,K)
       TB=B(J,J)*B(K,K)
       EPTOLB=TT/TB
       IF ((EPTOLA.LT.EPS).AND.(EPTOLB.LT.EPS)) GO TO 50
       AKK=A(K,K)*B(J,K)-B(K,K)*A(J,K)
       AJJ=A(J,J)*B(J,K)-B(J,J)*A(J,K)
       AB=A(J,J)*B(K,K)-A(K,K)*B(J,J)
       CHECK=(AB*AB+4.0*AKK*AJJ)/4.0
      IF (CHECK) 60,60,70
   60 WRITE (6,1004) CHECK
       STOP
   70 SQCH=SQRT(CHECK)
       D1=AB/2.0+SQCH
       D2=AB/2.0-SQCH
       DEN=D1
      IF (ABS(D2) .GT. ABS(D1)) DEN=D2
       IF (DEN) 90,80,90
 80    CA=0.
       CG=-A(J,K)/A(K,K)
       GO TO 100
```

```
 90     CA=AKK/DEN
        CG=-AJJ/DEN
C
C    WE PERFORM THE GENERALIZED ROTATION
 100    IF (N-2) 95,180,95
 95     JP1=J+1
        JM1=J-1
        KP1=K+1
        KM1=K-1
C
        IF (JM1-1) 120,110,110
 110    DO 105 I=1,JM1
        AJ=A(I,J)
        BJ=B(I,J)
        AK=A(I,K)
        BK=B(I,K)
        A(I,J)=AJ+CG*AK
        B(I,J)=BJ+CG*BK
        A(I,K)=AK+CA*AJ
 105    B(I,K)=BK+CA*BJ
C
 120    IF (KP1-N) 130,130,140
 130    DO 125 I=KP1,N
        AJ=A(J,I)
        BJ=B(J,I)
        AK=A(K,I)
        BK=B(K,I)
        A(J,I)=AJ+CG*AK
        B(J,I)=BJ+CG*BK
        A(K,I)=AK+CA*AJ
 125    B(K,I)=BK+CA*BJ
C
 140    IF (JP1-KM1) 150,150,180
 150    DO 160 I=JP1,KM1
        AJ=A(J,I)
        BJ=B(J,I)
        AK=A(I,K)
        BK=B(I,K)
        A(J,I)=AJ+CG*AK
        B(J,I)=BJ+CG*BK
        A(I,K)=AK+CA*AJ
 160    B(I,K)=BK+CA*BJ
 180    AK=A(K,K)
        BK=B(K,K)
        A(K,K)=AK+2*CA*A(J,K)+CA*CA*A(J,J)
        B(K,K)=BK+2*CA*B(J,K)+CA*CA*B(J,J)
        A(J,J)=A(J,J)+2*CG*A(J,K)+CG*CG*AK
        B(J,J)=B(J,J)+2*CG*B(J,K)+CG*CG*BK
        A(J,K)=0.0
        B(J,K)=0.0
C
C    UPDATE EIGENVECTORS
        DO 190 I=1,N
        XJ=X(I,J)
        XK=X(I,K)
```

```
      X(I,J)=XJ+CG*XK
 190  X(I,K)=XK+CA*XJ
C
 50   CONTINUE
C
      DO 220 I=1,N
 220  EIGV(I)=A(I,I)/B(I,I)
      IF(IFPR.EQ.0) GO TO 227
      WRITE (6,1005)
      WRITE (6,1002) (EIGV(I),I=1,N)
 227  CONTINUE
C
C     CHECK FOR CONVERGENCE
      DO 240 I=1,N
      TOL=RTOL*D(I)
      DIF=ABS(EIGV(I)-D(I))
      IF (DIF.GT.TOL) GO TO 300
 240  CONTINUE
C
C     CHECK IF ALL OFF-DIAG ELEMENTS ARE SATISFACTORILY SMALL
      EPS=RTOL**2
      DO 260 J=1,NR
      JJ=J+1
      DO 260 K=JJ,N
      TT=A(J,K)*A(J,K)
      TB=A(J,J)*A(K,K)
      EPSA=ABS(TT/TB)
      TT=B(J,K)*B(J,K)
      TB=B(J,J)*B(K,K)
      EPSB=TT/TB
      IF ((EPSA.LT.EPS).AND.(EPSB.LT.EPS)) GO TO 260
      GO TO 300
 260  CONTINUE
C
      DO 310 I=1,N
      DO 310 J=1,N
      B(J,I)=B(I,J)
 310  A(J,I)=A(I,J)
      RETURN
C
 300  DO 320 I=1,N
 320  D(I)=EIGV(I)
      IF (NSWEEP.LT.NSMAX) GO TO 40
      DO 330 I=1,N
      DO 330 J=1,N
      B(J,I)=B(I,J)
 330  A(J,I)=A(I,J)
      RETURN
C
 1000 FORMAT (27HOSWEEP NUMBER IN *JACOBI* =, I4)
 1002 FORMAT (1H ,12E11.4)
 1004 FORMAT (37HO***ERROR   SOLUTION STOP IN *JACOBI*, / 12X,
     1        8HCHECK = , E20.14 / 1X)
 1005 FORMAT (36HOCURRENT EIGENVALUES IN *JACOBI* ARE, / 1X)
C
```

```
      END
C
C     CALLED BY?  QTSHEL
C
C     THIS SUBROUTINE COMPUTES THE BENDING MOMENT FIELD IN A LCCT
C     PLATE BENDING ELEMENT WITH 6, 5, 4 OR 3 NODAL POINTS
C
C
C * * * * * * * * * * * * * * *  INPUTS  * * * * * * * * * * * * * * *
C
C  M,A,B,C,H  AS IN SLCCT.
C   W(I)      I=1...3   CORNER Z-DISPLACEMENTS
C RX(I)/RY(I) I=1...3   CORNER X/Y ROTATIONS
C   RM(I)     I=1...3   IF(M.GT.O), MIDPOINT SIDE ROTATIONS (DEVIATIONS
C             FROM NORMAL SLOPE LINEARITY)
C
C * * * * * * * * * * * * * *  OUTPUTS  * * * * * * * * * * * * * * * *
C
C   BMT(I,J)  I=1...3, J=1...3   BENDING MOMENT COMPONENTS  MOM-XX
C             (J=1), MOM-YY (J=2), AND MOM-XY (J=3) AT THE CORNERS
C             I=1...3  ASSOCIATED WITH THE INPUT DISPLACEMENTS
C
C
      SUBROUTINE LCTMOM (M)
      COMMON /TRIARG/ A(3),B(3),HMT(3),H(3),C(3,3),SMT(3,3),
     1 BMT(3,3),FT(12),W(3),RX(3),RY(3),RM(3),ST(12,12)
      DIMENSION U(3),Q(3,6),CV(3),IPERM(3),TX(3),TY(3),RH(3)
      EQUIVALENCE  (Q(1),ST(1))
      DATA  IPERM /2,3,1/
      AREA = A(3)*B(2)-A(2)*B(3)
      DO 150 I=1,3
      J = IPERM(I)
      X = A(I)**2+B(I)**2
      U(I)  = -(A(I)*A(J)+B(I)*B(J))/X
      X =SQRT(X)
      Y = 2.*AREA/X
      RH(I) = 0.0
      IF(I.LE.M) RH(I)  = 2.*Y*RM(I)
      TX(I)  =  Y*A(I)/X
      TY(I)  = -Y*B(I)/X
      A1 = A(I)/AREA
      A2 = A(J)/AREA
      B1 = B(I)/AREA
      B2 = B(J)/AREA
      Q(1,I)  = B1*B1
      Q(2,I)  = A1*A1
      Q(3,I)  = 2.* A1*B1
      Q(1,I+3)  = 2.* B1*B2
      Q(2,I+3)  = 2.* A1*A2
  150 Q(3,I+3)  = 2.*(A1*B2+A2*B1)
      DO 300 I=1,3
      J = IPERM(I)
      K = IPERM(J)
      FAC = H(I)**3/ 12.0
      A2 = A(J)
```

```
      A3 = A(K)
      B2 = B(J)
      B3 = B(K)
      U2 = U(J)
      U3 = U(K)
      W2 = 1.0 - U2
      W3 = 1.0 - U3
      C21 = -(2.0+W2)* B2 -  (2.0+U3)* B3 + TX(K)  - TX(J)
      C22 =    B2* W2 - B3* U3             + TX(K)  + TX(J)
      C31 = -(2.0+W2)* A2 -  (2.0+U3)* A3 + TY(K)  - TY(J)
      C32 =    A2* W2 - A3* U3             + TY(K)  + TY(J)
      C52 =    B2- B3* W3                  - TX(K)
      C62 =    A2- A3* W3                  - TY(K)
      C82 =    B2* U2- B3                  - TX(J)
      C92 =    A2* U2- A3                  - TY(J)
      C51 =    4.0* B3 - C52
      C61 =    4.0* A3 - C62
      C81 =   -4.0* B2 - C82
      C91 =   -4.0* A2 - C92
      DO 250 L=1,3
      Q11 = Q(L,I)
      Q22 = Q(L,J)
      Q33 = Q(L,K)
      Q12 = Q(L,I+3)
      Q23 = Q(L,J+3)
      Q31 = Q(L,K+3)
      Q1  = Q22 - Q33
      Q2  = Q22 - Q23
      Q3  = Q33 - Q23
      Q4  = Q23 + Q1
      Q5  = Q23 - Q1
  250 CV(L) =   (-6.*Q11+3.*((U3-W2)*Q1+(U3+W2)*Q23))*W(I)
     1        + (6.*Q22+3.*W3*Q4)*W(J)  +  (6.*Q33+3.*U2*Q5)*W(K)
     2        +((C21*Q1+C22*Q23+4.*(B2*Q31-B3*Q12))*RX(I)
     3        + (C31*Q1+C32*Q23+4.*(A2*Q31-A3*Q12))*RY(I)
     4        + (C51*Q22+C52*Q3)*RX(J)  +  (C61*Q22+C62*Q3)*RY(J)
     5        + (C81*Q33+C82*Q2)*RX(K)  +  (C91*Q33+C92*Q2)*RY(K)
     6        +  Q5*RH(J)  + Q4*RH(K))/2.
      DO 300 J=1,3
      BMT(I,J)  = -FAC*(C(J,1)*CV(1)+C(J,2)*CV(2)+C(J,3)*CV(3))
  300 CONTINUE
      RETURN
      END
      SUBROUTINE LCT9ST (SLCT9,NODE,XLCT9)
C
C     CALLED BY?  STRETR
C
C     THIS SUBROUTINE FORMS THE MOMENT RESULTANT/DISPLACEMENT TRANSFOR-
C     MATION MATRIX FOR A 3 NODE, LCCT-9, TRIANGULAR BENDING ELEMENT.
C
C**** I N P U T S
C
C     A,B,C,H        AS IN SLCCT.
C
C     NODE           NODE (1,2 OR 3) AT WHICH THE MOMENT/DISPLACEMENT
```

```
C                    IS FORMED
C
C**** O U T P U T S
C
C      SLCT9(I,J)    I=1...3, J=1...9.   MOMENT RESULTANTS AT TRIANGLE
C                    VERTEX (NODAL POINT) NUMBER  *NODE* ...     M(XX)/
C                    (I=1), M(YY)/(I=2), M(XY)/(I=3).  TRANSVERSE PLATE
C                    DISPLACEMENTS  W(1)/(J=1), W(2)/(J=2), W(3)/(J=3),
C                    AND IN-PLANE ROTATIONS  RX(1)/(J=4), RX(2)/(J=5),
C                    RX(3)/(J=6), RY(1)/(J=7), RY(2)/(J=8), RY(3)/(J=9).
C
      COMMON /TRIARG/
     1 A(3),B(3),HMT(3),H(3),C(3,3),SMT(3,3),BMT(3,3),
     1 FT(12),U(3),TX(3),TY(3),RM(3),ST(12,12)
C
      DIMENSION     Q(3,6),IPERM(3),SLCT9(3,9),XLCT9(3,9)
C
      EQUIVALENCE  (Q(1),ST(1))
C
      DATA          IPERM/2,3,1/
C
      AREA = A(3)* B(2) - A(2)* B(3)
C
      DO 150 I=1,3
      J = IPERM(I)
      X = A(I)**2 + B(I)**2
      U(I) = -(A(I)* A(J) + B(I)* B(J))/ X
      X =SQRT(X)
      Y = 2.0* AREA/ X
      TX(I) =  Y* A(I)/ X
      TY(I) = -Y* B(I)/ X
      A1 = A(I)/ AREA
      A2 = A(J)/ AREA
      B1 = B(I)/ AREA
      B2 = B(J)/ AREA
      Q(1,I  ) =        B1* B1
      Q(2,I  ) =        A1* A1
      Q(3,I  ) = 2.0*  A1* B1
      Q(1,I+3) = 2.0*  B1* B2
      Q(2,I+3) = 2.0*  A1* A2
      Q(3,I+3) = 2.0* (A1* B2+ A2* B1)
  150 CONTINUE
C
      I = NODE
      J = IPERM(I)
      K = IPERM(J)
      FAC = H(I)**3/ 12.0
      A2 = A(J)
      A3 = A(K)
      B2 = B(J)
      B3 = B(K)
      U2 = U(J)
      U3 = U(K)
      W2 = 1.0 - U2
      W3 = 1.0 - U3
```

```
C
      C21 = -(2.0+W2)* B2 -  (2.0+U3)* B3 + TX(K)  - TX(J)
      C22 =    B2* W2 - B3* U3            + TX(K)  + TX(J)
      C31 = -(2.0+W2)* A2 -  (2.0+U3)* A3 + TY(K)  - TY(J)
      C32 =    A2* W2 - A3* U3            + TY(K)  + TY(J)
      C52 =    B2- B3* W3                 - TX(K)
      C62 =    A2- A3* W3                 - TY(K)
      C82 =    B2* U2- B3                 - TX(J)
      C92 =    A2* U2- A3                 - TY(J)
      C51 =    4.0* B3 - C52
      C61 =    4.0* A3 - C62
      C81 =   -4.0* B2 - C82
      C91 =   -4.0* A2 - C92
C
      DO 250 L=1,3
      Q11 = Q(L,I)
      Q22 = Q(L,J)
      Q33 = Q(L,K)
      Q12 = Q(L,I+3)
      Q23 = Q(L,J+3)
      Q31 = Q(L,K+3)
      Q1  = Q22 - Q33
      Q2  = Q22 - Q23
      Q3  = Q33 - Q23
      Q4  = Q23 + Q1
      Q5  = Q23 - Q1
C     CURVATURE - DISPLACEMENT RELATION
      XLCT9(L,I)    = -6.*Q11+3.*((U3-W2)*Q1+(U3+W2)*Q23)
      XLCT9(L,J)    =  6.*Q22+3.*  W3*Q4
      XLCT9(L,K)    =  6.*Q33+3.*  U2*Q5
      XLCT9(L,I+3)  = (C21*Q1+C22*Q23+4.*(B2*Q31-B3*Q12))* 0.5
      XLCT9(L,J+3)  = (C51*Q22+C52*Q3)                   * 0.5
      XLCT9(L,K+3)  = (C81*Q33+C82*Q2)                   * 0.5
      XLCT9(L,I+6)  = (C31*Q1+C32*Q23+4.*(A2*Q31-A3*Q12))* 0.5
      XLCT9(L,J+6)  = (C61*Q22+C62*Q3)                   * 0.5
      XLCT9(L,K+6)  = (C91*Q33+C92*Q2)                   * 0.5
  250 CONTINUE
C     MOMENT - DISPLACEMENT RELATION
      DO 300 I=1,3
      DO 290 J=1,9
      DUM = 0.0
      DO 280 K=1,3
  280 DUM = DUM + C(I,K)* XLCT9(K,J)
      SLCT9(I,J)  = -FAC* DUM
  290 CONTINUE
  300 CONTINUE
C
      RETURN
      END
      SUBROUTINE LOAD (KTYPEE,PRR,YREFF,NFACE)
C
C     CALLS?  DERIV
C     CALLED BY?  BRICK8
C
      COMMON/EM/LM(24),ND,NS, ES(24,24),RF(24,4),XM(24),SA(12,24),
```

```
     .    SF(12,4),IFILL2(3048)
      COMMON /JUNK/ ETA(3)   ,DET,MLD(4),KLD(4),MULT(4),NP(8),INP(8),
     .               A(3,3),P(3,11),B(3,3),XX(8,3),Q(11),DL(8),IFLL(206)
      DIMENSION KTYPEE(1),PRR(1),YREFF(1),NFACE(1)
      COMMON /GASS / DUM(12),XK(4),DDUM(12),WGT(4),IPERM(3)
      DIMENSION KCRD(6),FVAL(6),KFACE(6,4)
C
      DATA KFACE /  1, 4, 2, 1, 6, 2,
     1              2, 3, 3, 4, 7, 3,
     2              6, 7, 7, 8, 8, 4,
     3              5, 8, 6, 5, 5, 1/
      DATA KCRD / 1,1,2,2,3,3/
      DATA FVAL /1.,-1.,1.,-1.,1.,-1./
C
C
      DO 700 KK=1,4
      NNN=KLD(KK)
      IF(NNN) 700,700,10
   10 KTYPE=KTYPEE(NNN)
      PR=PRR(NNN)
      YREF=YREFF(NNN)
      KF=NFACE(NNN)
C
C         INTEGRATE OVER THE SURFACE
C
      ML = KCRD(KF)
      MM = IPERM(ML)
      MN = IPERM(MM)
      ETA(ML)  = FVAL(KF)
      DO 300  LX = 1,4
      ETA(MM)  = XK(LX)
      DO 300  LY = 1,4
      ETA(MN)  = XK(LY)
      CALL DERIV(3,SA)
C
C         COMPUTE DIRECTION COSINES OF NORMAL TO SURFACE
C
      A1 =  (A(MM,2)*A(MN,3)-A(MM,3)*A(MN,2))
      A2 =  (A(MM,3)*A(MN,1)-A(MM,1)*A(MN,3))
      A3 =  (A(MM,1)*A(MN,2)-A(MM,2)*A(MN,1))
      AA=SQRT(A1**2+A2**2+A3**2)
      A1 = A1/AA
      A2 = A2/AA
      A3 = A3/AA
C
C         COMPUTE FIRST FUND. FORM   (SIN / )
C
      AA = 0.
      BB = 0.
      CC = 0.
      DO 200  I = 1,3
      AA=AA+A(MM,I)**2
      CC=CC+A(MN,I)**2
  200 BB = BB + A(MM,I)*A(MN,I)
      C=SQRT(AA*CC - BB*BB)
```

```
C
C        COMPUTE PRESSURE,LOAD COMPONENTS, STORE IN R
C
      IF (KTYPE.EQ.2) GO TO 170
      FORCE = PR
      GO TO 185
  170 YY = 0.
      DO 180  I = 1,8
  180 YY = YY + Q(I)*XX(I,2)
      YY = YY - YREF
      FORCE = -PR*YY
      IF(YY.GT.0.) FORCE = 0.
  185 CONTINUE
      TS=FORCE*WGT(LX)*WGT(LY)*C
C
      DO 190  I = 1,4
      N = KFACE(KF,I)
      QQ=TS*Q(N)
      K=3*N
      RF(K-2,KK)  = RF(K-2,KK)  + QQ*A1
      RF(K-1,KK)  = RF(K-1,KK)  + QQ*A2
      RF(K   ,KK)  = RF(K   ,KK)  + QQ*A3
  190 CONTINUE
C
  300 CONTINUE
C
  700 CONTINUE
C
      RETURN
      END
      SUBROUTINE LOADV (NLP,P,B,FF,IFF,LDOF,NEQ,NFN,KN)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE COMPUTES THE SYSTEM LOAD VECTORS *B* AT EACH OF THE
C     *NT* SOLUTION TIME STEPS AND SAVES THEM ON TAPE2.
C
      DIMENSION      NLP(NFN),P(KN,1),B(NEQ),FF(NEQ,NFN),IFF(NEQ,NFN),
     1               LDOF(NEQ)
C
      COMMON /DYN/   NT,NOT,ALFA,DT,BETA,IFILL(4)
C
C     READ FUNCTION MULTIPLIERS AND ARRIVAL TIME STEPS.  THESE ARRAYS
C     (*FF* AND *IFF*) ARE OVER-WRITTEN WITH LOAD VECTORS *B* IN THIS
C     ROUTINE.
C
      KT=2
      REWIND KT
      READ (KT) FF,IFF
      REWIND KT
C
      TETA=1.4
      DEL=TETA*DT-DT
C
C     SCAN THE FORCING FUNCTION MULTIPLIERS FOR ALL DEGREES OF FREEDOM
```

```
C     TO ELIMINATE THOSE DOF*S WHICH ARE UNLOADED.  ALSO DELETE THOSE
C     DOF*S WHOSE FUNCTIONS ARRIVE AFTER THE END OF THE SOLUTION PERIOD.
C
      KLOAD = 0
      DO 30 K=1,NEQ
      B(K) = 0.0
      DUM = 0.0
      IDUM = 0
      DO 20 I=1,NFN
      IF(IFF(K,I).GT.NT) GO TO 20
      IDUM = IDUM +1
   20 DUM = DUM +ABS(FF(K,I))
      IF(DUM.LT.1.0E-8) GO TO 30
      IF(IDUM.LT.1)     GO TO 30
      KLOAD = KLOAD +1
      LDOF(KLOAD) = K
   30 CONTINUE
      IF(KLOAD.GT.0) GO TO 40
      WRITE (6,3000)
 3000 FORMAT (32H0*** ERROR   SOLUTION TERMINATED, /
     1        13X,35HNO FORCES APPLIED TO THE STRUCTURE., / 1X)
      STOP
   40 CONTINUE
C
C     GENERATE SYSTEM LOADS AT EACH TIME STEP
C
      TT = 0.0
C
      DO 800 KK=1,NT
C
      TT = TT+DT
C
C     CONSIDER EACH LOADED DEGREE OF FREEDOM
C
      DO 700 KD=1,KLOAD
      KEQ = LDOF(KD)
      B(KEQ) = 0.0
C
C     CONSIDER EACH FORCING FUNCTION APPLIED TO THIS DEGREE OF FREEDOM
C
      DO 600 KF=1,NFN
C
C     PASS IF ZERO MULTIPLIER
C
      IF (ABS(FF(KEQ,KF)) .LT. 1.0D-8) GO TO 600
C
C     PASS IF THIS FUNCTION ARRIVES LATER THAN CURRENT TIME STEP, *KK*
C
      I = IFF(KEQ,KF) -1
      IF(I.GT.KK) GO TO 600
C
C     COMPUTE RELATIVE TIME TO BE USED FOR FUNCTION INTERPOLATION
C
      TR = TT - FLOAT(I)* DT
C
```

```
C     PASS IF THE FINAL TIME POINT IN THIS TABLE IS LESS THAN THE
C     RELATIVE TIME
C
      J = NLP(KF)
      TF = P(2*KF-1,J)
      IF(TF.LT.TR) GO TO 600
C
C     INTERPOLATE IN THIS TABLE FOR THE VALUE OF FUNCTION NUMBER *KF*
C     AT TIME *TR+DEL*
C
      NF2 = 2*KF
      NF1 = NF2-1
C
      DO 500 L=2,J
      IF(TR.GT.P(NF1,L)) GO TO 500
      RT = P(NF1,L)-P(NF1,L-1)
      IF(RT.GT.1.0E-8) GO TO 490
      M = L-1
      WRITE (6,3010) M,L,KF
 3010 FORMAT (53H0*** ERROR   ZERO OR NEGATIVE TIME DIFFERENCE BETWEEN,
     1        9H POINTS (,I3,7H) AND (,I3,1H), / 13X,8HFUNCTION,
     2        9H NUMBER (,I3,1H), / 1X)
      STOP
  490 RF = P(NF2,L)-P(NF2,L-1)
C
C     EXTRAPOLATE AN AMOUNT *DEL* BEYOND TIME *TR*
C
      FV = P(NF2,L-1) + (TR-P(NF1,L-1)+DEL)* RF/ RT
      GO TO 510
  500 CONTINUE
C
C     COMPUTE VALUE OF FORCE (APPLIED LOAD) AT THIS TIME STEP
C
  510 B(KEQ) = B(KEQ) + FF(KEQ,KF)* FV
C
  600 CONTINUE
  700 CONTINUE
C
C     SAVE THE LOAD VECTOR AT THIS STEP
C
      WRITE (KT) B
C
  800 CONTINUE
C
      RETURN
      END
      SUBROUTINE LOAD1(ID,FF,IFF,NUMNP,NEQB,NFN)
C
C     CALLED BY?  HISTRY
C
      DIMENSION ID(NUMNP,6),FF(NEQB,NFN),IFF(NEQB,NFN)
      COMMON / JUNK / NARB,NGM,IFILL1(428)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
C     READ ARBITRARY DYNAMIC LOADS
```

```
C
      IF(MODEX.EQ.1) GO TO 11
      NT=2
      REWIND NT
      REWIND 8
      READ (8) ID
C
      NNN=NEQB*NFN
      DO 10 I=1,NNN
      IFF(I,1)=0
   10 FF(I,1)=0.0D0
C
   11 WRITE (6,2000)
      NARB=1
   12 READ (5,1000) NP,IC,IFN,IAT,P
      IF(IAT.EQ.0) IAT=1
      IF(NP.GT.0) GO TO 15
      NARB=0
      RETURN
   15 WRITE (6,2002) NP,IC,IFN,IAT,P
      IF(MODEX.EQ.1) GO TO 12
C
      NS=1
      NE=NEQB
      DO 500 NN=1,NUMNP
      DO 500 II=1,6
C
   20 N=ID(NN,II)
      IF(N.LE.0) GO TO 300
      IF(N.GE.NS) GO TO 50
      WRITE (6,2001)
      STOP
C
   50 IF(N.LE.NE) GO TO 300
C
      WRITE (NT) FF,IFF
      NS=NS+NEQB
      NE=NE+NEQB
      IFF(I,1)=0
  100 FF(I,1)=0.0D0
C
      GO TO 50
C
  300 IF(NP.EQ.NN.AND.IC.EQ.II) GO TO 350
      GO TO 500
  350 M=N-NS+1
      IF(N.LE.0) GO TO 400
      FF(M,IFN)=P
      IFF(M,IFN)=IAT
  400 READ (5,1000) NP,IC,IFN,IAT,P
      IF(IAT.EQ.0) IAT=1
      WRITE(6,2002) NP,IC,IFN,IAT,P
      GO TO 300
C
  500 CONTINUE
```

```
      WRITE (NT) FF,IFF
C
      RETURN
C
 1000 FORMAT (4I5,F10.2)
 2001 FORMAT (18HODATA OUT OF ORDER )
 2000 FORMAT (29H1DYNAMIC NODAL FORCES/MOMENTS, // 14X,5HNODAL,7X,
     1 4HTIME,3X,7HARRIVAL,9X,4HTIME,/ 3X,4HNODE,3X,9HDEGREE OF,3X,
     2 8HFUNCTION,6X,4HTIME,5X,8HFUNCTION,/ 7H NUMBER,5X,7HFREEDOM,5X,
     3 6HNUMBER,4X,6HNUMBER,3X,10HMULTIPLIER, / 1X)
 2002 FORMAT (3X,I4,10X,I2,8X,I3,7X,I3,E13.4)
C
      END
      SUBROUTINE LOAD2 (FI,FF,IFF,PP,T,P,PD,NEQB,NF,NFN,NT,MAX,
     .                  NBLOCK,NAT)
C
C     CALLED BY?  HISTRY
C
      COMMON / EM / AT(1058),IFILL2(3022)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
      DIMENSION FI(NEQB,NF),FF(NEQB,NFN),IFF(NEQB,NFN),PP(NFN,1),T(1),
     .          P(1),PD(NT)
      COMMON / DYN /IQT,NOT,DAMP,DT,IFILL4(6)
      COMMON / JUNK / NARB,NGM,HED(12),IFILL1(404)
      COMMON /one/ A(1)
c***      COMMON A(7100)
C
C     TRANSFORM NODAL TO MODAL LOADS
C
      READ (5,1002) (AT(I),I=1,NAT)
      WRITE (6,2004) (I,AT(I),I=1,NAT)
      IF(MODEX.EQ.1) GO TO 302
      MT=4
      IF(NGM.EQ.0) MT=2
      REWIND MT
      NE=NAT*NF*NFN
      DO 10 I=1,NE
   10 A(I)=0.
      KK=NF*NFN
C
      DO 500 N=1,NBLOCK
      BACKSPACE 7
      READ (7) FI
      BACKSPACE 7
      READ (MT) FF,IFF
      NN=-KK
C
      DO 200 I=1,NF
      DO 200 J=1,NFN
      NN=NN+1
      DO 200 L=1,NEQB
      LL=IFF(L,J)
      IF(LL.EQ.0) GO TO 200
      K=NN+LL*KK
      A(K)=A(K) + FI(L,I)*FF(L,J)
```

```
  200 CONTINUE
  500 CONTINUE
C
C     READ TIME FUNCTIONS AND GENERATE
C     EQUAL INTERVAL FUNCTIONS
C
      TH=1.4
      DTA=DT*(TH - 1.)
  302 DO 335 I=1,NFN
C
      READ (5,1000) NLP,SFTR,HED
      IF(SFTR.EQ.0.) SFTR=1.0
      WRITE (6,2000) I,HED,NLP,SFTR
      IF(NLP.LE.MAX) GO TO 305
      L=2*(NLP-MAX)
      CALL ERROR(L)
  305 READ (5,1001)  (T(L),P(L),L=1,NLP)
      WRITE(6,2001)  (T(L),P(L),L=1,NLP)
      IF(MODEX.EQ.1) GO TO 335
C
      TIME=T(1)
      TIMEP=TIME + DTA
      L=1
      K=1
  310 L=L+1
      DDT=T(L)-T(L-1)
      DDP=P(L)-P(L-1)
      IF(DDT) 315,310,320
  315 WRITE (6,2003)
      STOP
  320 SLOPE=DDP/DDT
  323 IF  (T(L)-TIME) 310,325,325
  325 PP(I,K)=P(L-1) + (TIMEP - T(L-1))*SLOPE
      PP(I,K)=PP(I,K)*SFTR
  330 TIME=TIME+DT
      TIMEP=TIME + DTA
      K=K+1
      IF  (NT-K) 335,323,323
  335 CONTINUE
      IF(MODEX.EQ.1) RETURN
C
C     GENERATE MODAL LOAD VECTORS
C
      MT=4
      REWIND MT
      LL=NF*NFN
C
      DO 900 K=1,NF
      DO 550 I=1,NT
  550 PD(I)=0.
      INC=(K-1)*NFN
C
      DO 800 J=1,NAT
      LT=AT(J)/DT + 1
      N=0
```

```
C
      DO 600 NN=LT,NT
      N=N+1
      DO 600 I=1,NFN
      II=INC+I
  600 PD(NN)=PD(NN) + A(II)*PP(I,N)
C
  800 INC=INC+LL
C
  900 WRITE (MT) PD
C
      RETURN
C
 1000 FORMAT (I5,F10.0,12A5)
 1001 FORMAT (12F6.0)
 1002 FORMAT (8F10.2)
 2000 FORMAT (29H1TIME FUNCTION NUMBER      = (,I4,   1H), //
     1         29H FUNCTION DESCRIPTION     = (,12A5, 1H), //
     2         29H NUMBER OF ABSCISSAE      = (,I4,   1H), /
     3         29H FUNCTION SCALE FACTOR    = (,E13.4,1H), // 1X)
 2001 FORMAT (5(2X,10HTIME VALUE,4X,8HFUNCTION), / (5(F12.5,E12.4)))
 2003 FORMAT (38H0*** ERROR   TIME POINTS OUT OF ORDER., / 1X)
 2004 FORMAT (//// 20H ARRIVAL TIME VALUES, // 7H  ENTRY,3X,7HARRIVAL,
     1 5H TIME, / 7H NUMBER,10X,5HVALUE, // (I7,F15.6) )
      END
      SUBROUTINE LOSTR (IS,A,B,SA,SF,L)
C
C     CALLED BY?  BRICK8
C
      DIMENSION IS(2),A(3,3),B(3,3),SA(12,24),SF(12,4),IRF(6,2),TC(6,24)
     1,TR(6,6)
      DATA IRF /1,1,2,2,3,3,
     1          2,2,3,3,1,1/
C
      LL=IS(L)
      I=IRF(LL,1)
      TT=B(1,I)*B(1,I)+B(2,I)*B(2,I)+B(3,I)*B(3,I)
      TT=SQRT(TT)
      TC(3,1)=B(1,I)/TT
      TC(3,2)=B(2,I)/TT
      TC(3,3)=B(3,I)/TT
      I=IRF(LL,2)
      TT=A(I,1)*A(I,1)+A(I,2)*A(I,2)+A(I,3)*A(I,3)
      TT=SQRT(TT)
      TC(1,1)=A(I,1)/TT
      TC(1,2)=A(I,2)/TT
      TC(1,3)=A(I,3)/TT
      TC(2,1)=TC(3,2)*TC(1,3)-TC(3,3)*TC(1,2)
      TC(2,2)=TC(3,3)*TC(1,1)-TC(3,1)*TC(1,3)
      TC(2,3)=TC(3,1)*TC(1,2)-TC(3,2)*TC(1,1)
C
      TR(1,1)=TC(1,1)*TC(1,1)
      TR(1,2)=TC(1,2)*TC(1,2)
      TR(1,3)=TC(1,3)*TC(1,3)
      TR(1,4)=TC(1,1)*TC(1,2)*2.
```

```
      TR(1,5)=TC(1,2)*TC(1,3)*2.
      TR(1,6)=TC(1,1)*TC(1,3)*2.
      TR(2,1)=TC(2,1)*TC(2,1)
      TR(2,2)=TC(2,2)*TC(2,2)
      TR(2,3)=TC(2,3)*TC(2,3)
      TR(2,4)=TC(2,1)*TC(2,2)*2.
      TR(2,5)=TC(2,2)*TC(2,3)*2.
      TR(2,6)=TC(2,1)*TC(2,3)*2.
      TR(3,1)=TC(3,1)*TC(3,1)
      TR(3,2)=TC(3,2)*TC(3,2)
      TR(3,3)=TC(3,3)*TC(3,3)
      TR(3,4)=TC(3,1)*TC(3,2)*2.
      TR(3,5)=TC(3,2)*TC(3,3)*2.
      TR(3,6)=TC(3,1)*TC(3,3)*2.
      TR(4,1)=TC(1,1)*TC(2,1)
      TR(4,2)=TC(1,2)*TC(2,2)
      TR(4,3)=TC(1,3)*TC(2,3)
      TR(4,4)=TC(1,1)*TC(2,2)+TC(1,2)*TC(2,1)
      TR(4,5)=TC(1,2)*TC(2,3)+TC(1,3)*TC(2,2)
      TR(4,6)=TC(1,1)*TC(2,3)+TC(1,3)*TC(2,1)
      TR(5,1)=TC(2,1)*TC(3,1)
      TR(5,2)=TC(2,2)*TC(3,2)
      TR(5,3)=TC(2,3)*TC(3,3)
      TR(5,4)=TC(2,1)*TC(3,2)+TC(2,2)*TC(3,1)
      TR(5,5)=TC(2,2)*TC(3,3)+TC(2,3)*TC(3,2)
      TR(5,6)=TC(2,1)*TC(3,3)+TC(2,3)*TC(3,1)
      TR(6,1)=TC(3,1)*TC(1,1)
      TR(6,2)=TC(3,2)*TC(1,2)
      TR(6,3)=TC(3,3)*TC(1,3)
      TR(6,4)=TC(3,1)*TC(1,2)+TC(3,2)*TC(1,1)
      TR(6,5)=TC(3,2)*TC(1,3)+TC(3,3)*TC(1,2)
      TR(6,6)=TC(3,1)*TC(1,3)+TC(3,3)*TC(1,1)
C
      IL=(L-1)*6
      DO 100 I=1,6
      DO 100 J=1,24
      TC(I,J)=0.
      DO 100 K=1,6
  100 TC(I,J)=TC(I,J)+TR(I,K)*SA(IL+K,J)
      DO 110 I=1,6
      DO 110 J=1,24
  110 SA(IL+I,J)=TC(I,J)
C
      DO 120 I=1,6
      DO 120 J=1,4
      TC(I,J)=0.
      DO 120 K=1,6
  120 TC(I,J)=TC(I,J)+TR(I,K)*SF(IL+K,J)
      DO 130 I=1,6
      DO 130 J=1,4
  130 SF(IL+I,J)=TC(I,J)
C
      RETURN
      END
C
```

```
C     CALLED BY?  QTSHEL
C
C     THIS SUBROUTINE COMPUTES THE IN-PLANE STRESSES IN A LINEAR STRAIN
C     TRIANGLE (LST) WITH 6, 5 OR 4 NODAL POINTS, OR IN A CONSTANT
C     STRAIN TRIANGLE (CST)
C
C
C * * * * * * * * * * * * * * * INPUTS  * * * * * * * * * * * * * * * *
C
C  M,A,B,C     AS IN SLST.
C
C  U(I),V(I)   I=1...3+M   IN-PLANE NODAL DISPLACEMENT COMPONENTS.  THE
C              MIDPOINT VALUES U(I),V(I),I=4...3+M, IF ANY, ARE
C              DEVIATIONS FROM LINEARITY
C
C * * * * * * * * * * * * * * OUTPUTS  * * * * * * * * * * * * * * * *
C
C  SMT(I,J)    I=1...3, J=1...3   MEMBRANE STRESS COMPONENTS  SIG-XX
C              (J=1), SIG-YY (J=2), AND SIG-XY (J=3) AT THE CORNERS
C              I=1...3, ASSOCIATED WITH THE INPUT DISPLACEMENTS
C
C
      SUBROUTINE LSTSTR (M)
      COMMON /TRIARG/ A(3),B(3),H(3),HPT(3),C(3,3),SMT(3,3),
     1 BMT(3,3),U(6),V(6),W(3),RX(3),RY(3),RM(3),ST(12,12)
      DIMENSION  EXX(3),EYY(3),GXY(3),EPS(3,3),IPERM(3)
      EQUIVALENCE  (EXX(1),EPS(1)),(EYY(1),EPS(4)),(GXY(1),EPS(7))
      DATA  IPERM /2,3,1/
      AREA = A(3)*B(2)-A(2)*B(3)
      E11 =   (B(1)*U(1)+B(2)*U(2)+B(3)*U(3))/AREA
      E22 =   (A(1)*V(1)+A(2)*V(2)+A(3)*V(3))/AREA
      G12 =   (A(1)*U(1)+A(2)*U(2)+A(3)*U(3)+B(1)*V(1)+B(2)*V(2)
     1        +B(3)*V(3))/AREA
      DO 150 I=1,3
      EXX(I)  = E11
      EYY(I)  = E22
  150 GXY(I)  = G12
      IF(M.LE.0) GO TO 250
      DO 200 I=1,M
      X = 4.0*U(I+3)/AREA
      Y = 4.0*V(I+3)/AREA
      J = IPERM(I)
      K = IPERM(J)
      EXX(J)  = EXX(J)  + B(K)*X
      EXX(K)  = EXX(K)  + B(J)*X
      EYY(J)  = EYY(J)  + A(K)*Y
      EYY(K)  = EYY(K)  + A(J)*Y
      GXY(J)  = GXY(J)  + A(K)*X + B(K)*Y
  200 GXY(K)  = GXY(K)  + A(J)*X + B(J)*Y
  250 DO 300 I=1,3
      DO 300 J=1,3
  300 SMT(I,J)  = C(J,1)*EPS(I,1)+C(J,2)*EPS(I,2)+C(J,3)*EPS(I,3)
      RETURN
      END
      SUBROUTINE MODES (NEQ,MBAND,NBLOCK,NEQB,NF,MTOT,IFPR,IFSS,RTOL,
```

```
     1NITEM,COFQ)
C
C      CALLS?  SECNTD,SBLOCK,SSPCEB
C      CALLED BY?  SOLEIG
C
C    PROGRAM TO COMPUTE SMALLEST EIGENVALUES AND ASSOCIATED VECTORS IN
C    THE GENERALIZED EIGENVALUE PROBLEM
C                       A*V=RT*B*V  (A POS DEF,B DIAG NONNEG DEF)
C
C
       COMMON /SOL/ IDUM(5),NEIG,NAD,NVV,ANORM,NFO
        COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
        COMMON /one/ A(1)
c***        COMMON A(7100)
C
C
        NSTIF=4
        NMASS=9
        NRED=10
        NL=2
        NR=3
        NT=7
C
C      PRINT EIGENPROBLEM SUMMARY
C
       WRITE (6,1000) NEQ,MBAND,NBLOCK,NEQB,NF
C
        IF (NEIG.GT.0) GO TO 300
C
C      D E T E R M I N A N T   S E A R C H
C
       IF(NVV.GE.NF) GO TO 110
       WRITE (6,1010) NF,NVV
       STOP
  110 CONTINUE
        NIM=3
        NVM=6
        NC=NF+NIM
        NCA=NEQ*MAX0(MBAND,NC)
        N2=1+NCA
        N3=N2+NEQ
        N4=N3+NEQ
        N5=N4+NEQ
        N6=N5+NEQ
        N7=N6+NEQ*NVM
        N8=N7+NEQ*NVM
        N9=N8+NC
        N10=N9+NC
        N11=N10+NC
        N12=N11+NC
C
 200    CALL  SECNTD (A(1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),A(N9
     1),A(N10),A(N11),A(N12),NEQ,MBAND,NF,NC,IFPR,ANORM,COFQ)
        GO TO 600
C
```

```
C     S U B S P A C E   I T E R A T I O N
C
 300   NWA=NEQB*MBAND
       NV=2*NF
       IF (NF.GT.8) NV=NF+8
       IF (NAD.NE.0) NV=NAD
       IF (NVV.GE.NV) GO TO 310
       WRITE (6,1010) NV,NVV
       STOP
 310   NWV=NV*NEQB
       NTB=(MBAND-2)/NEQB+1
       IF (NTB.GE.NBLOCK) NTB=NBLOCK-1
       NWVV=NWV*(NTB+1)
C
C     CHECK FOR USE OF GIVEN STARTING ITERATION VECTORS
C
      IF (NFO.LE.0) GO TO 500
      REWIND 10
      READ (10) NEQO,NBLOKO,NEQBO,MBANDO,N10,NFO
      N2=1+NEQBO*NFO
      N3=N2+NEQB*NV
C
      CALL SBLOCK (A(1),A(N2),A(N3),NFO,NV,NEQBO,NEQB,NBLOKO,NBLOCK)
C
 500   CALL  SSPCEB (NEQ,MBAND,NBLOCK,NEQB,NF,NV,NWA,NWV,NWVV,NTB,IFPR,
     1IFSS,NITEM,RTOL,ANORM,COFQ)
C
 600   RETURN
C
 1000  FORMAT (//// 46H SOLUTION IS SOUGHT FOR FOLLOWING EIGENPROBLEM,//
     1          / 37H NUMBER OF EQUATIONS                  =,I5 //
     2            37H HALF BANDWIDTH OF STIFFNESS MATRIX =,I5 //
     3            37H NUMBER OF EQUATION BLOCKS           =,I5 //
     4            37H NUMBER OF EQUATIONS PER BLOCK       =,I5 //
     5            37H NUMBER OF EIGENVALUES REQUIRED      =,I5 // )
 1010 FORMAT (/// 32H0***ERROR   SOLUTION TERMINATED., /
     1          12X,40HNUMBER OF NON-ZERO MASSES REQUIRED     =, I5 /
     2          12X,40HNUMBER OF EXISTING MASSES IN THE MODEL =, I5 )
C
        END
       SUBROUTINE MULT (W,A,V,NN,MA)
C      CALLED BY ? SECNTD
C
C
       DIMENSION A(1),W(1),V(1)
C
       NM=NN*(MA -1)
       NMA=NN - MA + 1
       DO 20 I=1,NN
       W(I)=0.0
       K=I - 1
       IF (NMA -I) 10,15,15
    10 NM=NM - NN
    15 IL=NM + 1
       DO 20 J=I,IL,NN
```

```
      K=K + 1
   20 W(I)=W(I) + A(J)*V(K)
C
      IF (MA -1) 30,100,30
C
   30 KK=NN
      DO 40 I=2,MA
      II=I -1
      KK=KK + NN
      KJ=KK
      DO 40 J=1,II
      KJ=KJ -NN
   40 W(I)=W(I) + A(KJ + J)*V(J)
      IF (MA.EQ.NN) GO TO 100
      MA1=MA + 1
      IJ=1
      DO 50 I=MA1,NN
      KJ=KK
      IJ=IJ + 1
      II=I -1
      DO 50 J=IJ,II
      KJ=KJ - NN
   50 W(I)=W(I) + A(KJ + J)*V(J)
C
  100 RETURN
      END
      SUBROUTINE PINVER (A,NMAX,ND,NODE,NEL,MODEX)
C
C     CALLED BY?  TANGKS,BENDKS
C
C     INVERSION OF A POSITIVE DEFINITE MATRIX
C
      DIMENSION A(ND,ND)
C
      DO 200 N=1,NMAX
C
      IF (ABS(A(N,N)) .GT. 1.0D-20)  GO TO 50
      WRITE (6,2000) N,NODE,NEL
 2000 FORMAT (19H0*** ERROR.   ROW (,I2,27H) OF THE FLEXIBILITY MATRIX,
     1 10H AT NODE (,I4,20H) FOR PIPE ELEMENT (,I4,14H) IS SINGULAR., /
     2 1X )
      MODEX = 1
      RETURN
C
   50 D = 1.0/ A(N,N)
      DO 100 J=1,NMAX
  100 A(N,J) = -A(N,J) * D
C
      DO 150 I=1,NMAX
      IF(N.EQ.I) GO TO 150
      DO 140 J=1,NMAX
      IF(N.EQ.J) GO TO 140
      A(I,J) = A(I,J) + A(I,N)*A(N,J)
  140 CONTINUE
  150 A(I,N) = A(I,N)* D
```

```
C
      A(N,N) = D
C
  200 CONTINUE
C
      RETURN
      END
      SUBROUTINE PIPE
C
C     CALLS?  PIPEK,STRSC
C     CALLED BY?  ELTYPE
C
C     CONTROL ROUTINE FOR PIPE ELEMENT STIFFNESS/LOAD FORMATION AND
C     STRESS RECOVERY
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /JUNK/  LT,LH,L,IPAD1,SIG(18),N6,N7,N8,N9,N10,N11,N12,N13,
     1               N14,N15,N16,N17,N18,IPAD2,DUMMY(188)
      COMMON /EM/    NS,IFILL(5137)
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      DATA           HI/1HI/,HC/1HC/,HJ/1HJ/
C
      COMMON /one/         A(1)
C
      IF(NPAR(1).EQ.0) GO TO 500
C
C     FORM ELEMENT STIFFNESS, LOAD AND STRESS TRANSFORMATION MATRICES
C
C        1. ERROR CHECKS
C
      WRITE (6,2000)
      IF(NPAR(2).GT.0) GO TO 10
      WRITE (6,3000) (NPAR(K),K=1,7)
      WRITE (6,3010)
      STOP
   10 IF(NPAR(3).GT.0) GO TO 20
      WRITE (6,3000) (NPAR(K),K=1,7)
      WRITE (6,3020)
      STOP
   20 IF(NPAR(4).LT.1) NPAR(4) = 1
      IF(NPAR(5).GT.0) GO TO 30
      WRITE (6,3000) (NPAR(K),K=1,7)
      WRITE (6,3030)
      STOP
   30 IF(NPAR(6).LT.1) NPAR(6) = 0
      IF(NPAR(7).LT.3) NPAR(7) = 4
C
C        2. STORAGE ALLOCATION
C
      N6 = N5 + NUMNP
      KK = NPAR(4)* NPAR(3)
      N7 = N6 + KK
```

```
      N8 = N7 + KK
      N9 = N8 + KK
      N10= N9 + KK
      N11= N10+ NPAR(5)
      N12= N11+ NPAR(5)
      N13= N12+ NPAR(5)
      N14= N13+ NPAR(5)
      N15= N14+ NPAR(5)
      N16= N15+ NPAR(5)
      N17= N16+ NPAR(5)
      N18= N17+ NPAR(6)
      N19= N18+ NPAR(6)*NPAR(7)
C
      IF(N19.GT.MTOT) CALL ERROR (N19-MTOT)
C
C     DUMP FLAGS (GT.0, DUMP)
C
C     NPAR(9)  =  TANGENT ELEMENT MATRICES
C     NPAR(10) =  BEND ELEMENT MATRICES
C     NPAR(11) =  ELEMENT PROPERTIES
C     NPAR(12) =  NOT USED
C     NPAR(13) =  LOAD CASE NUMBER FOR WHICH ELEMENT FORCES ARE TO
C                 BE SAVED ON PUNCH CARDS
C     NPAR(14) =  5 DIGIT IDENTIFIER PUNCHED ON EACH ELEMENT FORCE CARD
C                 (APPEARS IN CC 76-80 ON EVERY CARD OUTPUT)
C
C
      CALL PIPEK (NPAR(2),NPAR(3),NPAR(4),NPAR(5),NPAR(6),NPAR(7),
     1            A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),
     2           A(N9),A(N10),A(N11),A(N12),A(N13),A(N14),A(N15),
     3            A(N16),A(N17),A(N18),NUMNP,MBAND)
C
      RETURN
C
C     COMPUTE ELEMENT STRESS OUTPUT
C
  500 CONTINUE
      WRITE (6,4000)
      LINE = 5
      NUMEL = NPAR(2)
C
      DO 800 MM=1,NUMEL
C
      CALL STRSC (A(N1),A(N3),NEQ,0)
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) NS
      DO 700 L=LT,LH
C
      CALL STRSC (A(N1),A(N3),NEQ,1)
      IF(LINE.LE.55) GO TO 510
      WRITE (6,4000)
      LINE = 5
  510 IF(NS.GT.12) GO TO 520
      WRITE (6,4010) MM,L,(SIG(I),I=1,12)
```

```
      LINE = LINE +3
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,(SIG(I),I=1,12)
      IF(NPAR(13).NE.L) GO TO 700
C     SAVE TANGENT FORCES/MOMENTS (I,J) FOR THIS LOAD CASE ON PUNCH
C     WRITE (11,5000) MM,(SIG(I),I=1,6),HI,L,NPAR(14)
C     WRITE (11,5000) MM,(SIG(I),I=7,12),HJ,L,NPAR(14)
      GO TO 700
  520 CONTINUE
      WRITE (6,4020) MM,L,(SIG(I),I=1,18)
      LINE = LINE +4
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,(SIG(I),I=1,18)
      IF(NPAR(13).NE.L) GO TO 700
C     SAVE BEND   FORCES/MOMENTS (C,J) FOR THIS LOAD CASE ON PUNCH
C     WRITE (11,5000) MM,(SIG(I),I=7,12),HC,L,NPAR(14)
C     WRITE (11,5000) MM,(SIG(I),I=13,18),HJ,L,NPAR(14)
C
  700 CONTINUE
  800 CONTINUE
C
      RETURN
C
 2000 FORMAT (46H1P I P E   E L E M E N T   I N P U T   D A T A, ///
     1 38H C O N T R O L   I N F O R M A T I O N, // 1X)
C
 3000 FORMAT ('63H ERROR DETECTED WHILE PROCESSING MASTER ELEMENT
     1 CONTROL CARD.', // 16X,1H(,7I5,1H), / 1X)
 3010 FORMAT (16X,26HNO PIPE ELEMENTS SPECIFIED, / 1X)
 3020 FORMAT (16X,22HNO MATERIALS REQUESTED, / 1X)
 3030 FORMAT (16X,31HNO SECTION PROPERTIES REQUESTED, / 1X)
C
 4000 FORMAT (46H1P I P E   F O R C E S   A N D   M O M E N T S, //
     1 8H ELEMENT,2X,7HELEMENT,3X,4HLOAD,2X,7HSTATION,8X,5HAXIAL,7X,
     2 6HY-AXIS,7X,6HZ-AXIS,4X, 9HTORSIONAL,7X,6HY-AXIS,7X,6HZ-AXIS, /
     3 2X,6HNUMBER,5X,4HTYPE,3X,4HCASE,17X,5HFORCE, 2(8X,5HSHEAR),
     4 3(7X,6HMOMENT), / 1X)
 4010 FORMAT (4X,I4,2X,7HTANGENT,4X,I3,4X,5HEND-I,3F13.3,3F13.2 / 28X,
     1          5HEND-J,3F13.3,3F13.2 / 1X)
 4020 FORMAT (4X,I4,5X,4HBEND,   4X,I3,4X,5HEND-I,3F13.3,3F13.2 / 27X,
     1          6HCENTER,3F13.3,3F13.2 / 28X,5HEND-J,3F13.3,3F13.2 / 1X)
C
 5000 FORMAT (3X,I3,4X,6E10.3,A1,I2,2X,I5)
C
      END
C
C     CALLS?  PIPES2,TANGDC,SELECT,TANGKS,BENDDC,PIPES3,BENDKS,CALBAN
C     CALLED BY?  PIPE
C
C     FORMATION OF 3-D PIPE TANGENT AND BEND ELEMENT STIFFNESS, LOAD
C     AND STRESS TRANSFORMATION ARRAYS
C
C     NPIPE        =  NUMBER OF PIPE ELEMENTS
```

```
C       NUMMAT         =  NUMBER OF MATERIALS
C       MAXTP          =  MAXIMUM NUMBER OF TEMPERATURE POINTS DESCRIBING
C                         ANY ONE MATERIAL
C       NPROP          =  NUMBER OF SECTION PROPERTY SETS
C       NBPN           =  NUMBER OF BRANCH POINT NODES
C       MAXTAN         =  MAXIMUM NUMBER OF TANGENT ELEMENTS COMMON TO A
C                         BRANCH POINT NODE
C       ID             =  MASTER INDEX ARRAY
C       X,Y,Z          =  NODE COORDINATES
C       T              =  NODE TEMPERATURES
C       TM             =  MATERIAL TEMPERATURE ARRAY
C       E              =  YOUNG*S MODULUS TABLE
C       XNU            =  POISSON*S RATIO TABLE
C       ALP            =  THERMAL EXPANSION COEFFICIENTS TABLE
C       DOUT           =  OUTSIDE DIAMETER OF THE PIPE SECTION
C       WALL           =  PIPE WALL THICKNESS
C       ALFAV          =  SHEAR SHAPE FACTOR
C       XWGT           =  SECTION WEIGHT PER UNIT LENGTH
C       XMAS           =  SECTION MASS   PER UNIT LENGTH
C       AREA           =  AREA OF THE PIPE CROSS SECTION
C       XMI            =  SECTION MOMENT OF INERTIA
C       NODBR          =  BRANCH POINT NODE ARRAY
C       NEBR           =  ARRAY CONTAINING TANGENT ELEMENT NUMBERS COMMON
C                         TO THE BRANCH NODE.  POSITIVE ELEMENT NUMBERS ARE
C                         ATTACHED TO THE BRANCH AT THEIR I-TH END.
C       NUMNP          =  NUMBER OF NODES
C       MBAND          =  EQUATION BANDWIDTH
C       S              =  ELEMENT STIFFNESS MATRIX
C       LM             =  ELEMENT EQUATION NUMBER ARRAY
C       RF             =  GLOBAL LOADS FOR EACH ELEMENT LOAD CASE
C       XM             =  ELEMENT LUMPED MASS MATRIX
C       SA             =  STRESS DISPLACEMENT TRANSFORMATION MATRIX
C       SF             =  RESTRAINED NODE CORRECTIONS FOR EACH ELEMENT
C                         LOAD CASE (A, B, C, OR D)
C
      SUBROUTINE PIPEK (NPIPE,NUMMAT,MAXTP,NPROP,NBPN,MAXTAN,
     1                  ID,X,Y,Z,T,TM,E,
     2                  XNU,ALP,DOUT,WALL,ALFAV,XWGT,XMAS,
     3                  AREA,XMI,NODBR,NEBR,NUMNP,MBAND)
      COMMON /PIPEC/ SHEAR,YM,POS,PARA1,PARA2,NOAX,NODE,NELEMT,MODEX,
     1               PARA3,THERM,PRESS,SECTA,SECTI,SECTW,SECTD,SECTM
      COMMON /EM/    LM(12),ND,NS,S(12,12),RF(12,4),XM(12),SA(18,12),
     1               SF(18,4),FEF(12,5),FEFC(18,5),FLEX(6,6),BT(6,6),
     2               HT(6,6),DC(3,3),IFILL3(3606)
      COMMON /JUNK/  TITLE(8),DC1(3,3),X1P(3),X2P(3),X3P(3),EMUL(5,4),
     1               HD(6),Q(3,3),QQ(3,3),FAC(5),AC(3),KB(2),HEDBR(8),
     2               IFILL1(256)
      COMMON /ELPAR/ NPAR(14),IFILL2(10)
      COMMON /EXTRA/ KODEX,NT8,IFILL4(14)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      DIMENSION      ID(NUMNP,1),X(1),Y(1),Z(1),T(1),TM(MAXTP,1),
     1               E(MAXTP,1),XNU(MAXTP,1),ALP(MAXTP,1),DOUT(1),
     2               WALL(1),ALFAV(1),XWGT(1),XMAS(1),AREA(1),XMI(1),
```

```
     3                NODBR(1),NEBR(NBPN,1)
C
      DIMENSION       HED1(2,2)
C
      DATA HED1      /6HTANGEN,6HBEND  ,1HT,1H /
      DATA HD1,HD2,HD3/5H NONE,5H AT I,5H AT J/
      DATA TG1,TG2,TG3,TG4,TG5/'B','I',' ','C','CC'/
C
C     INITIALIZATION
C
      WRITE (6,2000) NPIPE,NUMMAT,MAXTP,NPROP,NBPN,MAXTAN
      NOAX = 0
      IF(NPAR(8).LT.1) GO TO 5
      NOAX = 1
    5 WRITE (6,2005) NOAX
      MODEX = KODEX
      PI = 4.0D0*ATAN(1.0D0)
      ND = 12
C
C     READ AND PRINT MATERIAL PROPERTY DATA
C
      WRITE (6,2010)
      DO 45 K=1,NUMMAT
      READ (5,1000) N,NT,HD
      IF(NT.LT.1) NT = 1
      WRITE (6,2020) N,NT,HD
      IF(N.LE.NUMMAT) GO TO 10
      WRITE (6,3000)
      STOP
   10 IF(N.GT.0) GO TO 15
      WRITE (6,3010)
      STOP
   15 IF(NT.LE.MAXTP) GO TO 20
      WRITE (6,3020) MAXTP
      STOP
   20 CONTINUE
      IF(MAXTP.LT.2) GO TO 30
      DO 25 L=2,MAXTP
   25 TM(L,N) = 0.0
   30 CONTINUE
      DO 40 I=1,NT
      READ  (5,1005) TM(I,N),E(I,N),XNU(I,N),ALP(I,N)
      WRITE (6,2030) I,TM(I,N),E(I,N),XNU(I,N),ALP(I,N)
      IF(I.LT.2) GO TO 40
      IF(TM(I,N).GT.TM(I-1,N)) GO TO 40
      WRITE (6,3030)
      STOP
   40 CONTINUE
      IF(NT.LT.MAXTP) TM(NT+1,N) = -10000.0
   45 CONTINUE
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     1WRITE (NT8) ((TM(I,K),E(I,K),XNU(I,K),ALP(I,K),I=1,MAXTP),
     2               K=1,NUMMAT)
C***
```

```
C
C     READ AND PRINT SECTION PROPERTY DATA
C
      WRITE (6,2040)
      DO 70 K=1,NPROP
      READ  (5,1010) N,X1,X2,X3,X4,X5,(HD(J),J=1,3)
      IF(X5.LT.1.0E-12) X5 = X4/ 386.4
      WRITE (6,2050) N,X1,X2,X3,X4,X5,(HD(J),J=1,3)
      IF(N.GT.0 .AND. N.LE.NPROP) GO TO 50
      WRITE (6,3040) N
      STOP
   50 DOUT(N)   = X1
      WALL(N)   = X2
      ALFAV(N)  = X3
      XWGT(N)   = X4
      XMAS(N)   = X5
      IF(DOUT(N).GT.1.0E-8) GO TO 55
      WRITE (6,3050) N
      STOP
   55 IF(WALL(N).GT.1.0E-8) GO TO 60
      WRITE (6,3060) N
      STOP
C
C     COMPUTE SECTION PROPERTIES
C
   60 CALL PIPES2 (X1,X2,X3,X4,X5,PI)
      AREA(N) = X4
      XMI(N)  = X5
      IF(ALFAV(N).LT.1.0E-8) ALFAV(N) = X3
   70 CONTINUE
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     1WRITE (NT8)  (DOUT(N),WALL(N),ALFAV(N),XWGT(N),XMAS(N),AREA(N),
     2              XMI(N),N=1,NPROP)
C***
C
C     READ AND PRINT BRANCH POINT NODES
C
      IF(NBPN.LT.1) GO TO 90
      WRITE (6,2060)
      WRITE (6,2070)
      READ  (5,1020)  (NODBR(K),K=1,NBPN)
      WRITE (6,2080)  (K,NODBR(K),K=1,NBPN)
C
C     TEST FOR ADMISSIBLE NODE NUMBERS
C
      DO 80 L=1,NBPN
      IF(NODBR(L).GT.0 .AND. NODBR(L).LE.NUMNP) GO TO 80
      WRITE (6,3070) L
      STOP
   80 CONTINUE
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8)  (NODBR(N),N=1,NBPN)
C***
```

```
C
      DO 85 I=1,NBPN
      DO 85 L=1,MAXTAN
      NEBR(I,L) = 0
   85 CONTINUE
C
   90 CONTINUE
C
C     READ AND PRINT ELEMENT LOAD CASE MULTIPLIERS
C
      WRITE (6,2090)
      READ  (5,1030) ((EMUL(I,J),J=1,4),I=1,5)
      WRITE (6,2100) ((EMUL(I,J),J=1,4),I=1,5)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) ((EMUL(I,J),I=1,5),J=1,4)
C***
C
C     READ AND PRINT ELEMENT DATA
C     PERFORM GENERATION FOR TANGENT ELEMENTS MISSING IN SEQUENCE
C
      WRITE (6,2110)
      LINE = 7
      XLN1 = 0.0
      TR1  = 0.0
      PR1  = 0.0
      TAVG1= 0.0
      MAT1 = 0
      IS1  = 0
      DO 95 I=1,2
      DO 95 J=1,2
   95 DC1(I,J) = 0.0
      NEL = 0
      L = 0
  100 KG = 0
      READ (5,1040) INEL,X1,INI,INJ,IMAT,ISP,TRI,PRI,X2,X3,X4,INC
      ITYP = 1
      IF(X1.EQ.TG1) ITYP = 2
      XTAG = TG2
C
C     BRANCH DEPENDING ON ELEMENT TYPE
C
      GO TO (110,300),ITYP
C
C     T A N G E N T   E L E M E N T
C
  110 IF(INC.EQ.0) INC = 1
  115 L = L+1
      KG = KG +1
      ML = INEL-L
      IF(ML) 120,125,130
  120 WRITE (6,3080) INEL
      STOP
  125 NEL = INEL
      NI = INI
```

```
      NJ = INJ
      MAT = IMAT
      IS = ISP
      TR = TRI
      PR = PRI
      AC(1)  = X2
      AC(2)  = X3
      AC(3)  = X4
      INK    = INC
      GO TO 135
  130 NEL = INEL-ML
      INK = 0
      XTAG = TG3
      NI = IN + KG* INCR
      NJ = JN + KG* INCR
  135 CONTINUE
      IF(LINE.LT.57) GO TO 140
      WRITE (6,2110)
      LINE = 7
  140 CONTINUE
      WRITE (6,2120) NEL,HED1(1,1),HED1(1,2),NI,NJ,MAT,IS,TR,PR,X2,X3,
     1               X4,INK,XTAG,XTAG
      LINE = LINE +1
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) NEL,ITYP,NI,NJ,MAT,IS,TR,PR
C***
C
C     TEST DATA INPUT FOR ADMISSIBILITY
C
      IF(NI.GT.0 .AND. NI.LE.NUMNP) GO TO 150
      N = NI
  145 WRITE (6,3090) N,NEL
      STOP
  150 IF(NJ.GT.0 .AND. NJ.LE.NUMNP) GO TO 155
      N = NJ
      GO TO 145
  155 IF(MAT.GT.0 .AND. MAT.LE.NUMMAT) GO TO 160
      WRITE (6,3100) MAT,NEL
      STOP
  160 IF(IS.GT.0 .AND. IS.LE.NPROP) GO TO 165
      WRITE (6,3110) IS,NEL
      STOP
  165 CONTINUE
C
C     DETERMINE IF THIS ELEMENT IS COMMON TO A BRANCH POINT
C
      IF(NBPN.LT.1) GO TO 1700
C
      KB(1)  = NI
      KB(2)  = NJ
      DO 1650 NEND=1,2
      DO 1620 K=1,NBPN
      IF(NODBR(K).NE.KB(NEND)) GO TO 1620
      KEL = NEL
```

```
      IF(NEND.EQ.2) KEL = -KEL
      LOC = K
      GO TO 1630
 1620 CONTINUE
      GO TO 1650
 1630 DO 1640 J=1,MAXTAN
      IF(NEBR(LOC,J).NE.0) GO TO 1640
      NEBR(LOC,J) = KEL
      GO TO 1650
 1640 CONTINUE
      WRITE (6,4020) MAXTAN,KB(NEND)
      MODEX = 1
      GO TO 1700
 1650 CONTINUE
 1700 CONTINUE
C
C     COMPUTE GEOMETRIC DATA FOR THE TANGENT ELEMENT
C
      X1P(1) = X(NI)
      X1P(2) = Y(NI)
      X1P(3) = Z(NI)
      X2P(1) = X(NJ)
      X2P(2) = Y(NJ)
      X2P(3) = Z(NJ)
C
      CALL TANGDC (NEL,X1P,X2P,AC,DC,MODEX,XLN)
C
C     SELECT PROPERTIES FROM THE MATERIAL TABLE
C
      TAVG = 0.5*(T(NI)+T(NJ))
C
      CALL SELECT (MAT,NEL,TAVG,TM,E,XNU,ALP,MAXTP,YM,POS,THERM)
C
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) XLN,DC,YM,POS,THERM
C***
      IF(MODEX.EQ.1) GO TO 510
C
C     TEST TO SEE IF NEW ELEMENT MATRICES ARE REQUIRED
C
      DUM =ABS(XLN1-XLN) +ABS(TR1-TR) +ABS(PR1-PR) +ABS(TAVG1-TAVG)
      IDUM = IABS(MAT1-MAT) + IABS(IS1-IS)
      DUM = DUM +DFLOAT(IDUM)
      IF(DUM.GT.1.0E-6) GO TO 180
      DO 170 I=1,2
      DO 170 J=1,2
      DU2 =ABS(DC1(I,J)-DC(I,J))
      DU3 =ABS(DC(I,J)*1.0E-6)
      IF(DU2.GT.DU3) GO TO 180
  170 CONTINUE
      GO TO 510
C
  180 XLN1 = XLN
      TR1  = TR
```

```
      PR1   = PR
      TAVG1= TAVG
      MAT1 = MAT
      IS1   = IS
      DO 185 I=1,2
      DO 185 J=1,2
  185 DC1(I,J) = DC(I,J)
C
C     GENERATE THE TANGENT ELEMENT STIFFNESS, LOAD AND STRESS MATRICES
C
      SHEAR  = ALFAV(IS)
      NODE   = NJ
      NELEMT = NEL
      PARA3  = XLN
      PRESS  = PR
      SECTA  = AREA(IS)
      SECTI  = XMI(IS)
      SECTW  = WALL(IS)
      SECTD  = DOUT(IS)
      SECTM  = XMAS(IS)
C****
      IF(NPAR(11).LT.1) GO TO 6710
      WRITE (6,5000) SHEAR,NODE,NELEMT,PARA3,PRESS,SECTA,SECTI,SECTW,
     1               SECTD,SECTM
      WRITE (6,5010) ((DC(I,J),J=1,3),I=1,3)
      WRITE (6,5020) TAVG,YM,POS,THERM
 6710 CONTINUE
C****
C
      CALL TANGKS
C
      NS = 12
      DELT = TAVG - TR
      WGT = XWGT(IS)
C
      GO TO 400
C
C     B E N D   E L E M E N T
C
  300 L = L+1
      CTAG = TG4
      IF(LINE.LT.57) GO TO 305
      WRITE (6,2110)
      LINE = 7
  305 CONTINUE
      WRITE (6,2125) INEL,HED1(2,1),HED1(2,2),INI,INJ,IMAT,ISP,TRI,PRI,
     1               XTAG,CTAG
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) INEL,ITYP,INI,INJ,IMAT,ISP,TRI,PRI
C***
      READ  (5,1050) RADIUS,P3T,X3P,TOL
      IF(TOL.LT.1.0E-8) TOL = 0.1
      WRITE (6,2130) RADIUS,P3T,X3P,TOL
      LINE = LINE +3
```

```
      KODE = 1
      IF(P3T.EQ.TG5 ) KODE = 2
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) RADIUS,KODE,X3P,TOL
C***
C
C     TEST INPUT DATA FOR ADMISSIBILITY
C
      K = NEL +1
      IF(INEL.EQ.K) GO TO 310
      WRITE (6,4000) INEL,K
      STOP
  310 NEL = INEL
      NI = INI
      NJ = INJ
      IF(INI.GT.0 .AND. INI.LE.NUMNP) GO TO 320
      N = INI
  315 WRITE (6,3090) N,INEL
      STOP
  320 IF(INJ.GT.0 .AND. INJ.LE.NUMNP) GO TO 330
      N = INJ
      GO TO 315
  330 IF(IMAT.GT.0 .AND. IMAT.LE.NUMMAT) GO TO 340
      WRITE (6,3100) IMAT,INEL
      STOP
  340 IF(ISP.GT.0 .AND. ISP.LE.NPROP) GO TO 350
      WRITE (6,3110) ISP,INEL
      STOP
  350 IF(RADIUS.GT.1.0E-8) GO TO 360
      WRITE (6,4010) INEL
      STOP
  360 CONTINUE
C
C     COMPUTE GEOMETRIC DATA FOR THE BEND ELEMENT
C
      X1P(1) = X(NI)
      X1P(2) = Y(NI)
      X1P(3) = Z(NI)
      X2P(1) = X(NJ)
      X2P(2) = Y(NJ)
      X2P(3) = Z(NJ)
      TOL = TOL*WALL(ISP)
C
      CALL BENDDC (INEL,INI,INJ,X1P,X2P,X3P,RADIUS,KODE,DC,MODEX,THETA,
     1             TOL,PI)
C
C     SELECT PROPERTIES FROM THE MATERIAL TABLE
C
      TAVG = 0.5*(T(INI)+T(INJ))
C
      CALL SELECT (IMAT,INEL,TAVG,TM,E,XNU,ALP,MAXTP,YM,POS,THERM)
C
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
```

```
      *WRITE (NT8) THETA,DC,YM,POS,THERM
C***
       IF(MODEX.EQ.1) GO TO 510
C
C      CALCULATE THE BEND FLEXIBILITY FACTOR
C
       CALL PIPES3 (WALL(ISP),RADIUS,DOUT(ISP),YM,PRI,XKP)
C
C      GENERATE THE ELEMENT STIFFNESS, LOAD AND STRESS MATRICES
C
       SHEAR  = ALFAV(ISP)
       NODE   = INJ
       NELEMT = INEL
       PARA1  = XKP
       PARA2  = THETA
       PARA3  = RADIUS
       PRESS  = PRI
       SECTA  = AREA(ISP)
       SECTI  = XMI(ISP)
       SECTW  = WALL(ISP)
       SECTD  = DOUT(ISP)
       SECTM  = XMAS(ISP)
C****
       IF(NPAR(11).LT.1) GO TO 6711
       WRITE (6,5100) SHEAR,NODE,NELEMT,PARA1,PARA2,PARA3,PRESS,SECTA,
      1              SECTI,SECTW,SECTD,SECTM
       WRITE (6,5110) ((DC(I,J),J=1,3),I=1,3)
       WRITE (6,5120) TAVG,YM,POS,THERM
 6711  CONTINUE
C****
C
       CALL BENDKS
C
       DELT = TAVG-TRI
       WGT  = XWGT(ISP)
       XLN1 = 0.0
       ML = 0
       NS = 18
C
C      TRANSFORM THE ELEMENT STIFFNESS MATRIX FROM LOCAL TO
C      GLOBAL COORDINATES
C
  400  CONTINUE
       DO 450 IR=1,10,3
       IRS = IR-1
       DO 440 IC=IR,10,3
       ICS = IC-1
C
       DO 410 I=1,3
       II = IRS+I
       DO 410 J=1,3
       JJ = ICS+J
       Q(I,J) = S(II,JJ)
  410  CONTINUE
C
```

```
      DO 420 I=1,3
      DO 420 J=1,3
      QQ(I,J) = 0.0
      DO 415 KN=1,3
      QQ(I,J) = QQ(I,J) + Q(I,KN)* DC(J,KN)
  415 CONTINUE
  420 CONTINUE
C
      DO 430 I=1,3
      II = IRS+I
      DO 430 J=1,3
      JJ = ICS+J
      S(II,JJ) = 0.0
      DO 425 KN=1,3
      S(II,JJ) = S(II,JJ) + DC(I,KN)* QQ(KN,J)
  425 CONTINUE
  430 CONTINUE
C
  440 CONTINUE
  450 CONTINUE
C
      DO 460 I=1,12
      DO 460 J=1,12
      S(J,I) = S(I,J)
  460 CONTINUE
C
C     FORM THE ELEMENT MATRICES ASSOCIATED WITH EACH OF THE FOUR (A,B,
C     C AND D) ELEMENT LOADING COMBINATIONS
C
      DO 500 LC=1,4
C
C        1. FORM THE PARTICIPATION FACTORS FROM THE FIVE TYPES OF
C           LOADING FOR THIS ELEMENT LOAD CASE
C
      DO 475 I=1,3
      FAC(I) = 0.0
      DO 470 J=1,3
  470 FAC(I) = FAC(I) + DC(J,I)* EMUL(J,LC)
      FAC(I) = FAC(I) * WGT
  475 CONTINUE
C
      FAC(4) = EMUL(4,LC)* DELT
      FAC(5) = EMUL(5,LC)
C
C        2. COMPUTE THE FORCES ACTING ON THE NODES IN THE LOCAL SYSTEM
C
      DO 485 I=1,ND
      RF(I,LC) = 0.0
      DO 480 J=1,5
      RF(I,LC) = RF(I,LC) - FEF(I,J)* FAC(J)
  480 CONTINUE
  485 CONTINUE
C
C        3. TRANSFORM THE LOCAL NODE FORCES TO THE GLOBAL SYSTEM
C
```

```
      DO 489 IR=1,10,3
      IRS = IR-1
C
      DO 486 I=1,3
      J = IRS+I
  486 Q(I,1) = RF(J,LC)
C
      DO 488 M=1,3
      J = IRS+M
      RF(J,LC) = 0.0
      DO 487 KN=1,3
      RF(J,LC) = RF(J,LC) + DC(M,KN) * Q(KN,1)
  487 CONTINUE
  488 CONTINUE
C
  489 CONTINUE
C
C        4. FORM THE FIXED-END STRESS RESULTANT CORRECTIONS
C
      DO 495 I=1,NS
      SF(I,LC) = 0.0
      DO 490 J=1,5
      SF(I,LC) = SF(I,LC) + FEFC(I,J)* FAC(J)
  490 CONTINUE
  495 CONTINUE
C
  500 CONTINUE
C
C     FORM THE ELEMENT EQUATION NUMBER ARRAY
C
  510 DO 515 K=1,6
      LM(K)    = ID(NI,K)
  515 LM(K+6)  = ID(NJ,K)
C
C     SAVE THE STIFFNESS AND APPLIED LOAD MATRICES FOR LATER ASSEMBLY
C
      CALL CALBAN (MBAND,NDIF,LM,XM,S,RF,ND,ND,NS)
C
C     SAVE THE STRESS RECOVERY INFORMATION
C
      WRITE (1)  ND,NS,(LM(I),I=1,ND),((SA(I,J),I=1,NS),J=1,ND),
     1           ((SF(I,J),I=1,NS),J=1,4)
C
C     CHECK FOR THE LAST ELEMENT
C
  520 IF(NPIPE-NEL) 120,600,530
  530 IF(ML.GT.0) GO TO 115
      IN = INI
      JN = INJ
      INCR = INC
      GO TO 100
C
  600 IF(NBPN.LT.1) RETURN
C
C     PRINT BRANCH POINT SUMMARY
```

```
C
      WRITE (6,2140)
      DO 620 K=1,NBPN
C
      DO 610 J=1,MAXTAN
      HEDBR(J) = HD1
      IF(NEBR(K,J).GT.0) HEDBR(J) = HD2
      IF(NEBR(K,J).LT.0) HEDBR(J) = HD3
  610 CONTINUE
C
      WRITE (6,2150) K,NODBR(K),(NEBR(K,L),HEDBR(L),L=1,MAXTAN)
  620 CONTINUE
C
      KODEX = MODEX
      RETURN
C
C     FORMAT STATEMENTS
C
 1000 FORMAT (2I5,6A6)
 1005 FORMAT (4F10.0)
 1010 FORMAT (I5,5F10.0,3A6)
 1020 FORMAT (10I5)
 1030 FORMAT (4F10.0)
 1040 FORMAT (I4,A1,4I5,5F10.0,I5)
 1050 FORMAT (F10.0,3X,A2,4F10.0)
C
 2000 FORMAT (7X,33HNUMBER OF PIPE ELEMENTS          =, I6 //
     1        7X,33HNUMBER OF MATERIAL SETS          =, I6 //
     2        7X,26HMAXIMUM NUMBER OF MATERIAL,            /
     3        7X,33HTEMPERATURE INPUT POINTS         =, I6 //
     4        7X,33HNUMBER OF SECTION PROPERTY SETS =, I6 //
     5        7X,33HNUMBER OF BRANCH POINT NODES     =, I6 //
     6        7X,26HMAXIMUM NUMBER OF TANGENTS,            /
     7        7X,33HCOMMON TO A BRANCH POINT         =, I6 // 1X)
 2005 FORMAT (7X,25HFLAG FOR NEGLECTING AXIAL,             /
     1        7X,33HDEFORMATIONS IN BEND ELEMENTS    =, I6 /
     2        7X,15H(EQ.1, NEGLECT), // 1X)
 2010 FORMAT (//48H M A T E R I A L   P R O P E R T Y   T A B L E S,/1X)
 2020 FORMAT (//'OMATERIAL NUMBER    = (',I4,1H),/
     1           '1OH NUMBER OF',                    /
     2           '23H TEMPERATURE POINTS = (',I4,1H),/
     3           '23H IDENTIFICATION     = (',6A6,1H),//
     4 '2X,5HPOINT,19X,7HYOUNG*S,3X, 9HPOISSON*S,5X,7HTHERMAL',/
     . ' number'
     5,'3X,11HTEMPERATURE,5X,7HMODULUS,7X,5HRATIO,3X,9HEXPANSION', / 1X)
 2030 FORMAT (I7,F14.2,F12.1,F12.3,E12.3)
 2040 FORMAT (44H1S E C T I O N   P R O P E R T Y   T A B L E, //
     1 8H SECTION,4X,7HOUTSIDE,8X,4HWALL,3X,12HSHAPE FACTOR,7X,
     2 7HWEIGHT/, 9X,5HMASS/, / 8H  NUMBER,3X,8HDIAMETER,3X,9HTHICKNESS,
     3 6X,9HFOR SHEAR,2(3X,11HUNIT LENGTH),3X,21HD E S C R I P T I O N,
     4 / 1X)
 2050 FORMAT (I8,F11.3,F12.4,F15.4,2E14.4,3X,3A6)
 2060 FORMAT (44H1B R A N C H   P O I N T   N O D E   L I S T, /// 1X)
 2070 FORMAT (7H BRANCH,5X,4HNODE,/2X,5HPOINT,3X,6HNUMBER, / 1X)
 2080 FORMAT (I7,I9)
```

```
 2090 FORMAT (///34H E L E M E N T   L O A D   C A S E, 3X,
     1 21HM U L T I P L I E R S, // 31X,6HCASE A,4X,6HCASE B,4X,
     . 6HCASE C
     2,4X,'CASE D '  )
 2100 FORMAT (5X,19HX-DIRECTION GRAVITY, 3X, 4F10.3 /
     1         5X,19HY-DIRECTION GRAVITY, 3X, 4F10.3 /
     2         5X,19HZ-DIRECTION GRAVITY, 3X, 4F10.3 /
     3         5X,19HTHERMAL  DISTORTION, 3X, 4F10.3, /
     4         5X,19HPRESSURE DISTORTION, 3X, 4F10.3, // 1X)
 2110 FORMAT (46H1P I P E   E L E M E N T   I N P U T   D A T A, // 1X,
     1 7HELEMENT,2X,7HELEMENT,2(2X,4HNODE),3X,5HMATL.,2X,7HSECTION,4X,
     2 9HREFERENCE,2X,8HINTERNAL,3X,17HD I R E C T I O N,3X,
     313HC O S I N E S,7X,4HNODE,2X,5HINPUT, / 2X,6HNUMBER,5X,4HTYPE,4X,
     4 2H-I,4X,2H-J,2X,6HNUMBER,3X,6HNUMBER,2X,11HTEMPERATURE,2X,
     5 8HPRESSURE,7X,5HA(YX),7X,5HA(YY),7X,5HA(YZ),2X,9HINCREMENT,4X,
     6 3HTAG,/ 51X,5H(BEND,6X,6H(THIRD,3X,4H(X3-,8X,4H(Y3-,8X,4H(Z3-,7X,
     7 5H(WALL,/ 52X,7HRADIUS),4X,6HPOINT),3(3X,9HORDINATE)),2X,
     8 9HFRACTION), / 1X)
 2120 FORMAT (3X,I5,2X,A6,A1,2I6,3X,I5,4X,I5,F11.2,F12.2,3F12.4,I6,
     1         10X,2A1)
 2125 FORMAT (3X,I5,2X,A6,A1,2I6,3X,I5,4X,I5,F11.2,F12.2,52X,2A1)
 2130 FORMAT (48X,1H(,F9.3,1H),4X,1H(,A2,1H),2X,3(1H(,F10.3,1H) ),1X,
     1         1H(,F8.4,1H), / 1X)
 2140 FORMAT (34H1B R A N C H   P O I N T   D A T A, // 7H BRANCH,4X,
     1 4HNODE, / 7H  POINT,2X,6HNUMBER,3X,21HC O N N E C T I O N S,
     2 6H . . ., / 1X)
 2150 FORMAT (I7,I8,8(3X,I6,A5) )
C
 3000 FORMAT (41H0ERROR***  MATERIAL NUMBER EXCEEDS TOTAL., / 1X)
 3010 FORMAT (44H0ERROR***  MATERIAL NUMBER IS LESS THAN ONE., / 1X)
 3020 FORMAT (52H0ERROR***  NUMBER OF TEMPERATURE POINTS EXCEEDS USER,
     1 10H MAXIMUM (,I4,2H)., / 1X)
 3030 FORMAT (50H0ERROR***  TEMPERATURES MUST BE INPUT IN ASCENDING,
     1 7H ORDER., / 1X)
 3040 FORMAT (27H0ERROR***  SECTION NUMBER (,I5, 9H) IS BAD., / 1X)
 3050 FORMAT (41H0ERROR***  ZERO O.D. FOR SECTION NUMBER (,I4,2H).,/ 1X)
 3060 FORMAT (41H0ERROR***  ZERO WALL FOR SECTION NUMBER (,I4,2H).,/1X)
 3070 FORMAT (25H0ERROR***  BRANCH POINT (,I4,21H) HAS AN ILLEGAL NODE,
     1 18H NUMBER REFERENCE., / 1X)
 3080 FORMAT (27H0ERROR***  ELEMENT NUMBER (,'I4,'21H) IS OUT OF
     1 SEQUENCE',/ 1X)
 3090 FORMAT (17H0ERROR***  NODE (,I4,14H) OF ELEMENT (,I4,4H) IS,
     1 9H ILLEGAL., / 1X)
 3100 FORMAT (28H0ERROR***  MATERIAL NUMBER (,I4,19H) GIVEN FOR ELEMENT,
     1 9H NUMBER (,I4,13H) IS ILLEGAL., / 1X)
 3110 FORMAT (36H0ERROR***  SECTION PROPERTY NUMBER (,I4,11H) GIVEN FOR,
     1 17H ELEMENT NUMBER (,I4,13H) IS ILLEGAL., / 1X)
C
 4000 FORMAT (25H0ERROR***  BEND ELEMENT (,I4,21H) IS OUT OF SEQUENCE.,
     1 / 11X, 31HEXPECT TO READ ELEMENT NUMBER (,I4,1H), / 1X)
 4010 FORMAT (47H0ERROR***  ZERO RADIUS GIVEN FOR BEND ELEMENT (,I4,
     1 2H)., / 1X)
 4020 FORMAT (22H0ERROR***  MORE THAN (,I4,22H) TANGENT ELEMENTS ARE,
     1 24H COMMON TO BRANCH NODE (,I4, 2H)., / 1X)
C
```

```
 5000 FORMAT (// 10H SHEAR    =, E13.4 /
     1          10H NODE J   =, I4    /
     2          10H ELEMENT  =, I4    /
     3          10H LENGTH   =, E13.4 /
     4          10H PRESSURE=, E13.4 /
     5          10H AREA     =, E13.4 /
     6          10H INERTIA  =, E13.4 /
     7          10H WALL     =, E13.4 /
     8          10H O.D.     =, E13.4 /
     9          10H MASS     =, E13.4 //)
 5010 FORMAT (// 18H DIRECTION COSINES, // (3F15.8) )
 5020 FORMAT (// 14H T(AVERAGE)    =, E13.4 /
     1          14H YOUNG*S MOD =, E13.4 /
     2          14H POISSON*S    =, E13.4 /
     3          14H THERMAL EXP =, E13.4 //)
 5100 FORMAT (// 10H SHEAR    =, E13.4 /
     1          10H NODE J   =, I4    /
     2          10H ELEMENT  =, I4    /
     3          10H KAPPA    =, E13.4 /
     4          10H THETA    =, E13.4 /
     5          10H RADIUS   =, E13.4 /
     6          10H PRESSURE=, E13.4 /
     7          10H AREA     =, E13.4 /
     8          10H INERTIA  =, E13.4 /
     9          10H WALL     =, E13.4 /
     A          10H O.D.     =, E13.4 /
     B          10H MASS     =, E13.4 //)
 5110 FORMAT (// 18H DIRECTION COSINES, // (3F15.8) )
 5120 FORMAT (// 14H T(AVERAGE)    =, E13.4 /
     1          14H YOUNG*S MOD =, E13.4 /
     2          14H POISSON*S    =, E13.4 /
     3          14H THERMAL EXP =, E13.4 //)
C
      END
      SUBROUTINE PLANE
C
C     CALLS?  PLNAX,STRSC
C     CALLED BY?  ELTYPE
C
      COMMON /one/ A(1)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/ NS,ND,B(42,63),TI(42,4),LM(63)
      COMMON /JUNK/  LT,LH,L,IPAD,SG(20),SIG(7),EXTRA(150),N6,N7,N8,N9,
     1           N10,N11,N12,IFILL(65)
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      DIMENSION STRLAB(5)
      DATA STRLAB/3HCEN,3HL-I,3HJ-K,3HI-J,3HK-L/
C

      IF(NPAR(1).EQ.0)  GO TO 200
      IF(NPAR(1).EQ.3)  NPAR(5)=2
      IF(NPAR(5).EQ.0)  WRITE (6,2000)
      IF(NPAR(5).EQ.1)  WRITE (6,2001)
```

```
      IF(NPAR(5).EQ.2) WRITE (6,2002)
      IF  (NPAR(1).EQ.3)  WRITE  (6,2003)
      IF  (NPAR(6).NE.0)  WRITE  (6,2004)
      IF(NPAR(3).EQ.0)  NPAR(3)=1
      IF(NPAR(4).EQ.0)  NPAR(4)=1
      N6=N5+NUMNP
      N7=N6+NPAR(3)
      N8=N7+NPAR(3)
      N9=N8+NPAR(3)
      N10=N9+NPAR(3)
      N11=N10+11*NPAR(4)*NPAR(3)
      N12=N11+240
      MM=N12+240-MTOT
      IF(MM.GT.0)  CALL ERROR(MM)
C
      CALL PLNAX(A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),
     1           A(N9),A(N10),NPAR(4),NUMNP,A(N11),A(N12))
C
      RETURN
C
  200 WRITE (6,2006)
      NUME=NPAR(2)
      DO 800 MM=1,NUME
      CALL STRSC(A(N1),A(N3),NEQ,0)
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) NS
      IF(NS.EQ.1) GO TO 800
      WRITE (6,3000) MM
      DO 700 L=LT,LH
      CALL STRSC(A(N1),A(N3),NEQ,1)
      ITAG = 0
  510 DO 600 KK=1,NS,4
      ITAG = ITAG + 1
      DO 520 I=1,4
      II=KK-1+I
  520 SIG(I)=SG(II)
      CC=(SIG(1)+SIG(2))/2.0
      BB=(SIG(1)-SIG(2))/2.
      CR=SQRT(BB**2+SIG(4)**2)
      SIG(5)=CC+CR
      SIG(6)=CC-CR
      SIG(7)=0.0
      IF((BB.EQ.0.0) .AND. (SIG(4).EQ.0.0)) GO TO 530
      SIG(7)=28.648*ATAN2(SIG(4),BB)
C***  STRESS PORTHOLE
  530 IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,(SIG(I),I=1,7)
  600 WRITE (6,3001) L,STRLAB(ITAG),(SIG(I),I=1,7)
      WRITE (6,3002)
  700 CONTINUE
  800 CONTINUE
      RETURN
 2000 FORMAT (22H1AXISYMMETRIC ANALYSIS )
 2001 FORMAT (22H1PLANE STRAIN ANALYSIS )
```

```
 2002 FORMAT (22H1PLANE STRESS ANALYSIS )
 2003 FORMAT (18H MEMBRANE ELEMENTS )
 2004 FORMAT (30H INCOMPATIBLE MODES SUPPRESSED )
 2006 FORMAT (54H1T W O - D I M E N S I O N A L   F I N I T E   E L E M,
     1         8H E N T S,/// 8X,32H1.  CENTROID STRESSES REFERENCED,
     2        26H TO LOCAL Y-Z COORDINATES.,/  8X, 12H2.  MID-SIDE,
     3        51H STRESSES ARE NORMAL AND PARALLEL TO ELEMENT EDGES.,
     4        // 1X)
 3000 FORMAT (10HOELEMENT (,I5,1H),// 2X,4HLOAD,2X,3HLOC,12X,3HS11,12X,
     1        3HS22,12X,3HS33,12X,3HS12,10X,5HS-MAX,10X,5HS-MIN,5X,
     2        5HANGLE, / 1X)
 3001 FORMAT (I6,2X,A3,6E15.5,F10.2)
 3002 FORMAT (1HO)
      END
c************ s8.frc
      SUBROUTINE PIPES2 (DOUT,WALL,ALFAV,AREA,XMI,PI)
C
C     CALLED BY?  PIPEK
C
C     SECTION PROPERTY COMPUTATIONS FOR PIPE SECTIONS
C
C     DOUT          =  OUTSIDE DIAMETER
C     WALL          =  WALL THICKNESS
C     ALFAV         =  SHAPE FACTOR FOR SHEAR DISTORTION
C     AREA          =  CROSS SECTIONAL AREA
C     XMI           =  SECTION PRINCIPAL MOMENT OF INERTIA
C
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      ROUT = DOUT * 0.5
      RIN = ROUT - WALL
      ROUT2 = ROUT**2
      RIN2 = RIN**2
C     AREA
      AREA = PI* (ROUT2 - RIN2)
C     MOMENT OF INERTIA
      XMI = 0.25* PI* (ROUT2**2 - RIN2**2)
C     SHAPE FACTOR
      IF(ALFAV.GT.99.99) RETURN
      DUM2 = 1.333333333333* (ROUT2* ROUT - RIN2* RIN)
      DUM3 = (ROUT2 + RIN2) * WALL
      IF(DUM3.LT.1.0E-8) STOP 701
      ALFAV = DUM2/ DUM3
C
      RETURN
      END
      SUBROUTINE PIPES3 (WALL,RAD,DOUT,E,P,XKP)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
C     CALLED BY?  PIPEK
C
C     CALCULATION OF PRESSURE DEPENDENT FLEXIBILITY FACTOR
C
C     WALL          =  WALL THICKNESS
```

```
C      RAD            =  RADIUS OF THE BEND
C      DOUT           =  OUTSIDE DIAMETER OF THE PIPE
C      E              =  YOUNG*S MODULUS
C      P              =  INTERNAL PRESSURE
C      XKP            =  FLEXIBILITY FACTOR
C      RM             =  MEAN RADIUS OF THE PIPE
C
      RM = (DOUT - WALL)* 0.5
      IF(RM.LT.1.0E-8) STOP 702
      H = WALL* RAD/ RM**2
      IF(H.LT.1.0E-8) STOP 703
      IF(E.LT.1.0E-8) STOP 704
      DUM = 6.0* P/ E/ H
      IF(WALL.LT.1.0E-8) STOP 705
      DUM2 = (RAD/ WALL)** 1.333333333333
      DUM = 1.0 + DUM* DUM2
      XKP = (1.65/ H)/ DUM
      IF(XKP.LT.1.0) XKP = 1.0
C
      RETURN
      END
      SUBROUTINE PLNAX (ID,X,Y,Z,T,NTC,WT,RO,WANG,E,NUMTC,NUMNP,B,BB)
C
C     CALLS?  ELAW,QUAD,VECTOR,CROSS,DOT,CALBAN
C     CALLED BY?  PLANE
C
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,1),NTC(1),WT(1),RO(1),WANG(1),
     1          E(NUMTC,11,1),T(1),B(20,12),BB(20,12)
      COMMON /ELPAR/ NPAR(14),NUMNN,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/    LM(12),S(12,12),P(12,4),XM(12),
     1 TI(20,4),IX(4),IE(5),NS,D(4,4),EMUL(4,5),RR(4),ZZ(4),H(6),HS(6),
     2 HT(6),HR(6),HZ(6),FAC,XMM,PRESS,   EE(10),TTI(4),PP(12,4),THICK
     3 ,TMP(4),TP(12),ALP(4),IFILL2(4236)
      COMMON /JUNK/ MAT,NT,TEMP,REFT,BETA,U(4),V(4),W(4),G(4),IFLL(390)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      NUME=NPAR(2)
      NUMMAT=NPAR(3)
      numee=nume
         neqq=neq
      WRITE (6,2000) (NPAR(M),M=2,6)
C
C     READ AND PRINT OF MATERIAL PROPERTIES
C
      DO 60 M=1,NUMMAT
      READ  (5,1010) MAT,NTC(MAT),WT(MAT),RO(MAT),WANG(MAT)
        IF  (NTC(MAT).EQ.0)   NTC(MAT)=1
      WRITE (6,2020) MAT,NTC(MAT),WT(MAT),RO(MAT),WANG(MAT)
      NT=NTC(MAT)
      READ  (5,1005) ((E(I,J,MAT),J=1,11),I=1,NT)
      WRITE (6,2010) ((E(I,J,MAT),J=1,11),I=1,NT)
   60 CONTINUE
C***  DATA PORTHOLE SAVE
```

```
      IF(MODEX.EQ.0) GO TO 75
      DO 70 M=1,NUMMAT
      WRITE (NT8) M,NTC(M),WT(M),WANG(M)
      NT = NTC(M)
      WRITE (NT8) ((E(I,J,M),J=1,11),I=1,NT)
   70 CONTINUE
   75 CONTINUE
C
C     ELEMENT LOAD CASE MULTIPLIERS
C
      READ (5,1002)  ((EMUL(I,J),J=1,5),I=1,4)
      WRITE (6,2004) ((EMUL(I,J),J=1,5),I=1,4)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) ((EMUL(I,J),J=1,5),I=1,4)
C
C       READ AND PRINT OF ELEMENT PROPERTIES
C
      WRITE (6,2002)
      N=0
  130 READ(5,1003) M,(IE(I),I=1,5),REFT,PRESS,NS,KG,THICK
      MAT=IE(5)
      IF(KG.EQ.0) KG=1
      IF (NPAR(5).EQ.1) THICK=1.0
      IF(NS.EQ.0) NS=4
      IF(NS.LT.4) NS=1
      IF( (IE(3).EQ.IE(4)).AND.(NS.EQ.20) ) NS=16
  140 N=N+1
      IF(M.EQ.N) GO TO 145
      DO 142 I=1,4
  142 IX(I)=IX(I)+KG
      GO TO 149
  145 DO 148 I=1,4
  148 IX(I)=IE(I)
C
C        FORM CONSTITUTIVE LAW AND COMPUTE THERMAL STRESSES
C
  149 NT=NTC(MAT)
      WRITE (6,2003) N,IX,MAT,REFT,PRESS,NS,KG,THICK
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.0) GO TO 150
      WRITE (NT8) N,IX,MAT,REFT,PRESS,NS,THICK
      GO TO 153
  150 CONTINUE
      I=IX(1)
      J=IX(2)
      K=IX(3)
      L=IX(4)
      TEMP = (T(I)+T(J)+T(K)+T(L))/4.0
      BETA=WANG(MAT)
      XMM=RO(MAT)
      WGT=WT(MAT)
      CALL ELAW (NUMTC,EE,E,D,TTI,ALP)
C
C     CALCULATE ELEMENT STIFFNESS MATRIX
```

```
C
  153 IF(NPAR(1).EQ.3) GO TO 160
      ND=8
      DO 155 I=1,4
      II=IX(I)
      RR(I)=Y(II)
      ZZ(I)=Z(II)
      TMP(I) = T(II)
      LM(I)=ID(II,2)
  155 LM(I+4)=ID(II,3)
      IF(MODEX.EQ.1) GO TO 300
C
      CALL QUAD (B,BB)
C
      DO 158 I=1,4
      DO 157 L=1,4
      P(I,L)=P(I,L)+XM(I)*WGT*EMUL(L,4)
  157 P(I+4,L)=P(I+4,L)+XM(I)*WGT*EMUL(L,5)
      XM(I)=XM(I)*XMM
  158 XM(I+4)=XM(I)
      GO TO 300
C
  160 ND = 12
      IF(MODEX.EQ.1) GO TO 165
      CALL VECTOR(V,X(I),Y(I),Z(I),X(J),Y(J),Z(J))
      CALL VECTOR(G,X(I),Y(I),Z(I),X(L),Y(L),Z(L))
      CALL CROSS(V,G,W)
      CALL CROSS(W,V,U)
      CALL VECTOR(W,X(I),Y(I),Z(I),X(K),Y(K),Z(K))
      RR(1)=0.0
      ZZ(1)=0.0
      RR(2)=V(4)
      ZZ(2)=0.0
      RR(3)=W(4)*DOT(W,V)
      ZZ(3)=W(4)*DOT(W,U)
      RR(4)=G(4)*DOT(G,V)
      ZZ(4)=G(4)*DOT(G,U)
C
  165 DO 170 I=1,4
      II=IX(I)
      TMP(I) = T(II)
      LM(I)=ID(II,1)
      LM(I+4)=ID(II,2)
  170 LM(I+8)=ID(II,3)
      IF(MODEX.EQ.1) GO TO 300
C
      CALL QUAD (B,BB)
C
      DO 190 I=1,3
      DO 190 K=1,4
      KK=4*(I-1)+K
      DO 180 L=1,4
  180 PP(KK,L)=V(I)*P(K,L)+U(I)*P(K+4,L)
      DO 190 J=1,3
      DO 190 L=1,4
```

```
      LL=4*(J-1)+L
  190 BB(KK,LL)=V(I)*(S(K,L)*V(J)+S(K,L+4)*U(J))
     1 +U(I)*(S(K+4,L)*V(J)+S(K+4,L+4)*U(J))
C
      DO 196 I=1,12
      DO 194 L=1,4
  194 P(I,L)=PP(I,L)
      DO 196 J=1,12
      S(I,J)=BB(I,J)
  196 S(J,I)=S(I,J)
C
      DO 210 K=1,NS
      DO 200 L=1,4
      DO 200 J=1,3
      LL=4*(J-1)+L
  200 BB(K,LL)=B(K,L)*V(J)+B(K,L+4)*U(J)
      DO 210 J=1,12
  210 B(K,J)=BB(K,J)
C
      DO 220 I=1,4
      DO 215 L=1,4
      P(I  ,L)=P(I  ,L)+XM(I)*WGT*EMUL(L,3)
      P(I+4,L)=P(I+4,L)+XM(I)*WGT*EMUL(L,4)
  215 P(I+8,L)=P(I+8,L)+XM(I)*WGT*EMUL(L,5)
      XM(I)=XM(I)*XMM
      XM(I+4)=XM(I)
  220 XM(I+8)=XM(I)
C
C     CALCULATION OF BAND WIDTH AND WRITES ELEMENT MATRICES ON TAPES
C
  300 CALL CALBAN (MBAND,NDIF,LM,XM,S,P,ND,12,NS)
      IF(MODEX.EQ.1) GO TO 310
      WRITE (1) ND,NS,(LM(I),I=1,ND),(( B(I,J),I=1,NS),J=1,ND),
     1 ((TI(I,J),I=1,NS),J=1,4)
  310 IF(N.EQ.NUME) RETURN
      IF(N.EQ.M) GO TO 130
      GO TO 140
C
 1002 FORMAT (5F10.0)
 1003 FORMAT (6I5,2F10.0,2I5,F10.0)
 1005 FORMAT (8F10.0/3F10.0)
 1010 FORMAT (2I5,3F10.0)
 2000 FORMAT (// 23H NUMBER OF ELEMENTS   =, I6 /
     1          23H NUMBER OF MATERIALS  =, I6 /
     2          23H MAXIMUM TEMPERATURES  ,    /
     3          23H PER MATERIAL         =, I6 /
     4          23H ANALYSIS CODE        =, I6 /
     5          23H CODE FOR INCLUSION    ,    /
     6          23H OF BENDING MODES     =, I6 /
     7          23H   EQ.O, INCLUDE       ,    /
     8          23H   GT.O, SUPPRESS      ,    //// 1X)
 2002 FORMAT (8H1ELEMENT,26X,4HMATL,5X,9HREFERENCE,3X,8HI-J FACE,3X,
     1        6HSTRESS, / 2X,6HNUMBER,5X,1HI,5X,1HJ,5X,1HK,5X,1HL,2X,
     2        4HTYPE,3X,11HTEMPERATURE,3X,8HPRESSURE,3X,6HOPTION,4X,
     3        2HKG,3X,9HTHICKNESS, / 1X)
```

```
 2003 FORMAT (I8,5I6,F14.3,E11.3,I9,I6,F12.4)
 2004 FORMAT (/// 25H ELEMENT LOAD MULTIPLIERS, // 10H LOAD CASE,4X,
     1         11HTEMPERATURE,3X,8HPRESSURE,3X,9HX-GRAVITY,3X,
     2         9HY-GRAVITY,3X,9HZ-GRAVITY, // 5X,1HA,F19.3,F11.3,3F12.3 /
     3         5X,1HB,F19.3,F11.3,3F12.3 /    5X,1HC,F19.3,F11.3,3F12.3 /
     4         5X,1HD,F19.3,F11.3,3F12.3  )
 2010 FORMAT (F12.2,3E12.4,3F9.4,E12.4,3E14.4)
 2020 FORMAT (/// 25H MATERIAL I.D. NUMBER    =, I5 /
     1              25H NUMBER OF TEMPERATURES =, I5 /
     2              25H WEIGHT DENSITY         =, E14.4 /
     3              25H MASS   DENSITY         =, E14.4 /
     4              25H BETA ANGLE             =, F9.3 //
     5         12H TEMPERATURE,8X,4HE(N),8X,4HE(S),8X,4HE(T),3X,6HNU(NS),
     6         3X,6HNU(NT),3X,6HNU(ST),7X,5HG(NS),6X,8HALPHA(N),6X,
     7         8HALPHA(S),6X,8HALPHA(T)   )
      END
      SUBROUTINE PLOAD(ID,FF,IFF,NUMNP,NEQ,NFN)
C
C     CALLED BY?  STEP
C
C     READ FORCING FUNCTION DATA.
C     TERMINATE READING WITH A ZERO NODE NUMBER ON INPUT.
C     STORE FUNCTION MULTIPLIERS IN *FF* AND ARRIVAL TIME REFERENCES
C     IN *IFF*.
C     SAVE  FF,IFF  ON TAPE2 FOR LATER USE IN LOAD VECTOR ASSEMBLY.
C
      COMMON /EXTRA/ MODEX,NT8,IFILL(14)
C
      DIMENSION ID(NUMNP,6),FF(NEQ,NFN),IFF(NEQ,NFN)
C
      IF(MODEX) 10,10,20
   10 REWIND 8
      READ  (8) ID
      GO TO 30
   20 REWIND 2
      READ  (2) ID
      GO TO 60
C
   30 NT=2
      REWIND NT
C
      DO 50 I=1,NEQ
      DO 50 J=1,NFN
      IFF(I,J)=1.0D0
   50 FF(I,J)=0.0D0
C
   60 WRITE (6,2000)
C
C     CARD READING LOOP
C
   75 READ (5,1000) NP,IC,IFN,IAT,P
      IF (NP.EQ.0)  GO TO 200
      IF (IAT.EQ.0)  IAT=1
      IF(IFN.LT.1) IFN = 1
      WRITE (6,2002) NP,IC,IFN,IAT,P
```

```
      IF(NP.LE.NUMNP) GO TO 80
      WRITE (6,3010) NP
      STOP
   80 IF(IC.GT.0 .AND. IC.LT.7) GO TO 82
      WRITE (6,3020) IC
      STOP
   82 IF(IFN.LE.NFN) GO TO 84
      WRITE (6,3030) IFN
      STOP
   84 CONTINUE
      N=ID(NP,IC)
      IF (N) 100,100,150
  150 IF(MODEX.EQ.1) GO TO 75
       FF(N,IFN)=P
      IFF(N,IFN)=IAT
      GO TO 75
  100 WRITE (6,3000)  NP,IC
      STOP
  200 IF(MODEX.EQ.1) RETURN
C
C     SAVE FUNCTION MULTIPLIERS AND ARRIVAL TIME REFERENCES
C
      WRITE (NT) FF,IFF
      RETURN
C
C     F O R M A T S
C
 1000 FORMAT (4I5,F10.2)
 2000 FORMAT (36H1D Y N A M I C   L O A D   I N P U T, // 3X,4HNODE,3X,
     1         9HDEGREE OF,3X,8HFUNCTION,3X,12HARRIVAL TIME,5X,
     2         8HFUNCTION,/ 7H NUMBER,5X,7HFREEDOM,2X,9HREFERENCE,9X,
     3         6HNUMBER,3X,10HMULTIPLIER, / 1X)
 2002 FORMAT (I7,7X,I5,6X,I5,10X,I5,E13.4)
 3000 FORMAT (46H0*** ERROR   LOAD APPLIED TO A CONSTRAINED DOF, /
     1         13X,6HNODE (,I5,14H)  COMPONENT (,I5,1H), / 1X)
 3010 FORMAT (19H0*** ERROR   NODE (,I5,15H) OUT OF RANGE., / 1X)
 3020 FORMAT (24H0*** ERROR   COMPONENT (,I5,13H) IS ILLEGAL., / 1X)
 3030 FORMAT (33H0*** ERROR   FUNCTION REFERENCE (,I5,9H) IS BAD., / 1X)
C
      END
      SUBROUTINE POSINV(A,NMAX,NDD)
C
C     CALLED BY?  ELAW
C
      DIMENSION A(NDD,NDD)
C
      DO 200 N=1,NMAX
C
      D=A(N,N)
      DO 100 J=1,NMAX
 IF(D.EQ.0.)D=0.005
  100 A(N,J)=-A(N,J)/D
C
      DO 150 I=1,NMAX
      IF(N-I) 110,150,110
```

```
  110 DO 140 J=1,NMAX
      IF(N-J) 120,140,120
  120 A(I,J)=A(I,J)+A(I,N)*A(N,J)
  140 CONTINUE
  150 A(I,N)=A(I,N)/D
C
      A(N,N)=1.0/D
C
  200 CONTINUE
C
      RETURN
      END
      SUBROUTINE PPLOT (IT,JT,NDS,ISP)
C
C     CALLED BY?  DISPLY
C
      COMMON /  EM  / PP(101),KD(3,8),XM(8),TM(8),IP(8),X(8),IFILL(4856)
      DIMENSION SM(8)
      DATA SM /1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8 /
      DATA BL /1H /,V /1HX/,AST /1H*/
      COMMON / DYN / NT,NOT,DAMP,DT,IFILL2(6)
C
C
      READ (IT) KD,XM,TM,L
      WRITE (6,3000) (KD(1,I),KD(2,I),XM(I),TM(I),I,I=1,L)
C
      DO 100 K=1,L
      TT = XM(K)
      IF(ABS(TT).GT.1.0E-8 ) XM(K) = 50.0/ TT
  100 CONTINUE
      TT=0.
      WRITE (6,999)
      WRITE (6,1000)
      WRITE (6,2000) TT,(V,I=1,101),TT
C
      K=1
      DO 200 I=2,100
  200 PP(I)=BL
C
      DO 500 N=1,NDS
      READ (JT) X
      PP(1)=V
      PP(51)=V
      PP(101)=V
C
  220 II=ISP
  210 IF(II.LE.0) GO TO 250
      WRITE (6,2001) PP
      II=II-1
      GO TO 210
C
  250 TT=TT+DT
      DO 300 I=1,L
      XX=XM(I)*X(I)
      M=XX
```

```
      M=M+51
      IP(I)=M
      IF(PP(M).EQ.V .OR. PP(M).EQ.BL) GO TO 270
      PP(M) = AST
      GO TO 300
  270 PP(M) = SM(I)
  300 CONTINUE
      IF(K.LT.10)  GO TO 320
      K=1
      WRITE (6,2000) TT,PP,TT
      GO TO 340
  320 WRITE (6,2001) PP
      K=K+1
C
C     RESET PP
  340 DO 360 I=1,L
      M=IP(I)
  360 PP(M)=BL
  500 CONTINUE
      TT=TT+DT
      WRITE (6,2000) TT,(V,I=1,101),TT
      WRITE (6,1000)
C
      RETURN
C
  999 FORMAT (1H1,57X,15HO R D I N A T E )
 1000 FORMAT ( / 1H ,3X,7HT I M E,2X,4H-1.0,21X,4H-0.5,22X,3H0.0,22X,
     1         3H0.5,22X,3H1.0,4X,7HT I M E, 1X)
 2000 FORMAT (1H ,F10.4,4X,101A1,F12.4)
 2001 FORMAT (1H ,14X,101A1)
 3000 FORMAT (I8,12X,I3,1P2E14.4,3X,I6)
      END
      SUBROUTINE PRIST (NS,IS1,IS2,SIG,SPR)
C
C     CALLED BY?  THREED
C
      DIMENSION SIG(12),SPR(6),IS(2),SG(6)
C
      IS(1)=IS1
      IS(2)=IS2
      NNS=1
      IF (NS.EQ.12) NNS=2
      DO 900 N=1,NNS
      IN=3*N-3
      II=IN*2
      IF (IS(N).EQ.0) GO TO 200
C
      CC=(SIG(II+1)+SIG(II+2))/2.
      BB=(SIG(II+1)-SIG(II+2))/2.
      CR=SQRT(BB**2+SIG(II+4)**2)
      SPR(IN+1)=CC+CR
      SPR(IN+2)=CC-CR
      SPR(IN+3)=0.
      IF (BB .NE. 0.)SPR(IN+3)=28.648*ATAN2(SIG(II+4),BB)
      GO TO 900
```

```
C
  200 CC=(SIG(II+1)+SIG(II+2)+SIG(II+3))/3.
      DO 210 I=1,3
      SG(I)=SIG(II+I)-CC
  210 SG(I+3)=SIG(II+I+3)
      C2=(SG(1)**2+SG(2)**2+SG(3)**2)*.5+SG(4)**2+SG(5)**2+SG(6)**2
      C3=SG(1)*(SG(2)*SG(3)-SG(5)*SG(5))+SG(4)*(SG(5)*SG(6)-SG(4)*SG(3))
     1+SG(6)*(SG(4)*SG(5)-SG(2)*SG(6))
      T=SQRT(C2/1.5)
      A=C3*1.414214/T**3
      IF (ABS(A) .GT. 1.) A=SIGN(1.,A)
C***      A=DARCOS(A)/3.0D0
      A=ACOS(A)/3.0D0
      T=T*1.414214
      SPR(IN+1)=T*COS(A)
      SPR(IN+2)=T*COS(A+2.0944)
      SPR(IN+3)=T*COS(A-2.0944)
      DO 220 I=2,3
      IF (SPR(IN+1).GT.SPR(IN+I)) GO TO 220
      C3=SPR(IN+1)
      SPR(IN+1)=SPR(IN+I)
      SPR(IN+I)=C3
  220 CONTINUE
      IF (SPR(IN+2).LE.SPR(IN+3)) GO TO 230
      C3=SPR(IN+2)
      SPR(IN+2)=SPR(IN+3)
      SPR(IN+3)=C3
  230 DO 240 I=1,3
  240 SPR(IN+I)=SPR(IN+I)+CC
  900 CONTINUE
C
      RETURN
      END
      SUBROUTINE QDCOS (N,X,Y,Z,T)
C
C     CALLED BY?  STRETR,QTSHEL
C
C     THIS SUBROUTINE COMPUTES THE DIRECTION COSINES OF THE LOCAL
C     ELEMENT SYSTEM OF A QUADRILATERAL (N=4) OR SINGLE TRIANGLE (N=1)
C
      DIMENSION X(1), Y(1), Z(1), T(1)
      X1 = X(2)+X(3)-X(N)-X(1)
      Y1 = Y(2)+Y(3)-Y(N)-Y(1)
      Z1 = Z(2)+Z(3)-Z(N)-Z(1)
      X2 = X(3)+X(N)-X(1)-X(2)
      Y2 = Y(3)+Y(N)-Y(1)-Y(2)
      Z2 = Z(3)+Z(N)-Z(1)-Z(2)
      S1 = X1**2+Y1**2+Z1**2
      C  = (X1*X2+Y1*Y2+Z1*Z2)/S1
      X2 = X2 - C*X1
      Y2 = Y2 - C*Y1
      Z2 = Z2 - C*Z1
      S1 =SQRT (S1)
      S2 =SQRT (X2**2+Y2**2+Z2**2)
      X1 = X1/S1
```

```
      Y1 = Y1/S1
      Z1 = Z1/S1
      X2 = X2/S2
      Y2 = Y2/S2
      Z2 = Z2/S2
      T(1) = X1
      T(2) = X2
      T(3) = Y1*Z2-Y2*Z1
      T(4) = Y1
      T(5) = Y2
      T(6) = Z1*X2-Z2*X1
      T(7) = Z1
      T(8) = Z2
      T(9) = X1*Y2-X2*Y1
      RETURN
      END
C
C     CALLS?  QDCOS,TDCOS,TRFPRD,SLST,LSTSTR,SLCCT,LCTMOM
C     CALLED BY?  TPLATE
C
C     THIS SUBROUTINE CAN EVALUATE
C            ... ELEMENT STIFFNESS MATRIX      ...
C            ... CONSISTENT NODAL FORCE VECTOR ...
C            ... INTERNAL STRESSES AND MOMENTS ...
C     OF A SHALLOW QUADRILATERAL SHELL ELEMENT ASSEMBLED WITH 4 FLAT
C     TRIANGLES, OR OF A SINGLE TRIANGULAR SHELL ELEMENT.
C
C
C * * * * * * * * * * * * CALLING ARGUMENTS * * * * * * * * * * * * * *
C
C     INPUTS
C
C      KKK        INTEGER FLAG SPECIFYING OPERATION TO BE PERFORMED
C                  IF KKK =-1, FORM STIFFNESS MATRIX ONLY.
C                  IF KKK = 0, FORM STIFFNESS MATRIX AND LOAD VECTOR.
C                  IF KKK = 1, FORM LOAD VECTOR ONLY.
C                  IF KKK = 2, EVALUATE STRESSES AND MOMENTS.
C
C      NNS        NUMBER OF SUPPLIED NODAL POINTS
C                  IF NNS = 5, QTSHEL FORMS A QUADRILATERAL, AND THE
C                  PROPERTIES AT THE INTERNAL NODE 5 MUST BE INPUT.
C                  IF NNS = 4, QTSHEL FORMS A QUADRILATERAL, AND THE
C                  PROPERTIES AT THE INTERNAL NODE 5 ARE SET BY QTSHEL
C                  TO BE THEIR CORNER AVERAGE.
C                  IF NNS = 3, QDSTIF FORMS A SINGLE TRIANGLE.
C
C      NPF        NUMBER OF GLOBAL DEGREES OF FREEDOM AT EACH
C                 EXTERNAL NODE (3, 5 OR 6)
C                  IF NPF = 6, THE 3 DISPLACEMENTS U, V AND W AND THE
C                  3 ROTATIONS RX, RY AND RZ ARE INCLUDED AS D.O.F.
C                  IF NPF = 5, THE ROTATION RZ IS IGNORED.
C                  IF NPF = 3, ONLY U, V AND W ARE CONSIDERED AND
C                  THE BENDING STIFFNESS IS NOT INCLUDED (MEMBRANE
C                  SHELL ELEMENT)
C
```

```
C       MID          NUMBER OF INTERNAL MIDPOINTS IN QUADRILATERAL (0 OR 4)
C                     IF MID = 0, THE MEMBRANE ELEMENTS ARE   CST
C                            AND THE BENDING  ELEMENTS ARE   LCCT-9
C                     IF MID = 4, THE MEMBRANE ELEMENTS ARE   LST-10
C                            AND THE BENDING  ELEMENTS ARE   LCCT-11
C                    IF NNS = 3 (SINGLE TRIANGLE) MID IS ASSUMED TO BE 0
C
C       IDIS         INTEGER FLAG FOR THE NODAL DISPLACEMENTS U,V,W
C                     IF IDIS = 0, U,V,W ARE SPECIFIED IN THE GLOBAL SYSTEM
C                     IF IDIS = 1, U,V,W ARE SPECIFIED IN THE NODAL DISPL
C                     SYSTEMS DEFINED BY THE DIRECTION COSINE ARRAY TDIS.
C
C       IROT         INTEGER FLAG FOR THE NODAL ROTATIONS RX,RY,RZ.
C                     IF IROT = 0, RX,RY,RZ ARE SPECIFIED IN GLOBAL SYSTEM
C                     IF IROT = 1, RX,RY,RZ ARE SPECIFIED IN THE NODAL ROT
C                     SYSTEMS DEFINED BY THE DIRECTION COSINE ARRAY TROT.
C
C      OUTPUTS
C
C       NEF          NUMBER OF EXTERNAL DEGREES OF FREEDOM (NEF = NPF*NEN,
C                    WHERE NEN=4 FOR QUADRILATERAL, =3 FOR SINGLE TRIANGLE)
C
C       NTF          TOTAL NUMBER OF DEGREES OF FREEDOM (EXTERNAL+INTERNAL)
C
C
C * * * * * * * * * * ARRAYS IN  COMMON /QTSARG/  * * * * * * * * * * *
C
C X(I),Y(I),Z(I)    I=1...NNS   GLOBAL NODAL COORDINATES
C
C     CM(I,J)       I=1...3, J=1...3   PLANE STRESS MATERIAL MATRIX
C                   RELATING STRESSES TO STRAINS IN THE LOCAL SYSTEM
C
C     ALFA(I)       I=1...3   DILATATION COEFFICIENTS RELATING IN-PLANE
C                   THERMAL STRAINS IN THE LOCAL SYSTEM TO TEMPERATURES
C
C      HM(I)        I=1...NNS   THICKNESS RESISTING MEMBRANE STRESSES
C
C      HP(I)        I=1...NNS   THICKNESS RESISTING BENDING MOMENTS
C
C     RHO(I,J)      I=1...NNS, J=1...3  GLOBAL COMPONENTS RHOX (J=1),
C                   RHOY (J=2) AND RHOZ (J=3) OF BODY FORCES PER UNIT
C                   OF VOLUME
C
C      HW(I)        I=1...NNS   THICKNESS FOR COMPUTING BODY FORCES
C                   RHO*HW PER UNIT OF ELEMENT AREA
C
C      P(I)         I=1...NNS   LATERAL PRESSURE (NORMAL TO THE FACES OF
C                   THE COMPONENT TRIANGLES)
C
C      T(I)         I=1...NNS   MEAN TEMPERATURE VARIATIONS
C
C      DT(I)        I=1...NNS   MEAN TEMPERATURE THICKNESS GRADIENTS
C
C     SM(I,J)       I=1...NNS, J=1...3   ARRAY OF MEMBRANE STRESS
C                   COMPONENTS IN THE LOCAL SYSTEM  SIG-XX (J=1), SIG-YY
```

```
C                    (J=2) AND SIG-XY (J=3) . SM CONTAINS
C                     MEMBRANE STRESSES IN THE INITIAL POSITION AS INPUT
C                     WHEN KKK=0,1,2 (EXCLUDING THERMAL ACTIONS)
C                     MEMBRANE STRESSES IN THE DEFORMED POSITION AS OUTPUT
C                     WHEN KKK=2 (INCLUDING THERMAL ACTIONS)
C
C     BM(I,J)        I=1...NNS, J=1...3   ARRAY OF BENDING MOMENT
C                    COMPONENTS IN THE LOCAL SYSTEM  MOM-XX (J=1), MOM-YY
C                    (J=2) AND MOM-XY (J=3) . BM CONTAINS
C                     BENDING MOMENTS IN THE INITIAL POSITION AS INPUTS
C                     WHEN KKK=0,1,2 (EXCLUDING THERMAL ACTIONS)
C                     BENDING MOMENTS IN THE DEFORMED POSITION AS OUTPUT
C                     WHEN KKK=2 (INCLUDING THERMAL ACTION)
C
C    TDIS(I,J,K)     I=1...3, J=1...3, K=1...NEN    NOT REQUIRED IF
C                    IDIS=0. IF IDIS=1, TDIS(1..3,1..3,K) MUST CONTAIN
C                    THE (3,3) DIRECTION COSINE MATRIX OF THE NODAL
C                    DISPLACEMENT SYSTEM AT THE K-TH ELEMENT NODE WITH
C                    RESPECT TO THE GLOBAL SYSTEM
C
C    TROT(I,J,K)     I=1...3, J=1...3, K=1...NEN    NOT REQUIRED IF
C                    IROT=0. IF IROT=1, TROT(1..3,1..3,K) MUST CONTAIN
C                    THE (3,3) DIRECTION COSINE MATRIX OF THE NODAL
C                    ROTATION SYSTEM AT THE K-TH ELEMENT NODE WITH
C                    RESPECT TO THE GLOBAL SYSTEM
C
C      S(I,J)        I=1...NEF, J=1...NEF  EXTERNAL STIFFNESS MATRIX
C                    (OUTPUT IF KKK=-1,0)
C
C      S(I,J)        I=1...NTF, J=NEF+1...NTF   REDUCED INTERNAL STIFFNESS
C                    OF QUADRILATERAL ELEMENT. OUTPUT IF KKK=-1,0.
C                    REQUIRED INPUT IF KKK=1,2. NOT USED FOR SINGLE
C                    TRIANGLE.
C
C       R(I)         I=1...NEF    OUTPUT EXTERNAL NODAL FORCES IF KKK=0,1.
C                    INPUT EXTERNAL NODAL DISPLACEMENTS IF KKK=2.
C
C       R(I)         I=NEF+1...NTF   REDUCED INTERNAL NODAL FORCE VECTOR
C                    OF QUADRILATERAL ELEMENT. OUTPUT IF KKK=0,1.
C                    REQUIRED INPUT IF KKK=2 (RETURNS INTERNAL NODAL
C                    DISPLACEMENTS). NOT USED FOR SINGLE TRIANGLE.
C
C
C * * * * * * * * * * ROLE OF ARRAYS IN COMMON /QTSARG/ * * * * * * * *
C
C       ARRAYS       OPERATION  KKK =-1    KKK = 0    KKK = 1    KKK = 2
C                                Q   T      Q   T      Q   T      Q   T
C
C   X,Y,Z,CM,ALFA,HM,HP  (*)     I   I      I   I      I   I      I   I
C      RHO,HW,P                  -   -      I   I      I   I      -   -
C        T,DT   (*)              -   -      I   I      I   I      I   I
C        SM,BM  (*)              -   -      I   I      I   I     I/O I/O
C      TDIS,TROT  (**)           I   I      I   I      I   I      I   I
C   S(1..NEF,1..NEF)             O   O      O   O      -   -      -   -
C  S(1..NTF,NEF+1..NTF)          O   -      O   -      I   -      I   -
```

```
C        R(1..NEF)                      -   -     0   0     0   0      I   I
C      R(NEF+1..NTF)                    -   -     0   -     0   -     I/0  -
C
C          WHERE   Q=QUADRILATERAL (NNS=4,5), T=SINGLE TRIANGLE (NNS=3)
C                  I=INPUT, O=OUTPUT, I/O=INPUT/OUTPUT, -=NOT USED.
C
C      NOTES    (*)  HP,DT AND BM ARE NOT USED IF NPF=3.
C               (**) TDIS IS NOT USED IF IDIS=0, AND TROT IS NOT USED
C                    IF IROT=0.
C
C
      SUBROUTINE QTSHEL (KKK,NNS,NPF,MID,IDIS,IROT,NEF,NTF)
      COMMON /QTSARG/ X(5),Y(5),Z(5), HM(5),HP(5), CM(3,3),ALFA(3),
     1 HW(5),RHO(5,3),P(5), T(5),DT(5), SM(5,3),BM(5,3), TDIS(36),
     2     TROT(36),S(30,30),R(30)
      COMMON /TRIARG/ A(3),B(3), HMT(3),HPT(3), C(3,3), SMT(3,3),
     1 BMT(3,3), FT(12),     P1(3),P2(3),P3(3),RM(3), ST(12,12)
      COMMON /TRANSF/  T1(3),T2(3),T3(3), T0(3,3)
      COMMON /EXTRA/ MODEX
      DIMENSION F(1), IPERMQ(4), MFR(5), LOC(5), NC(3), CA(3), WGT(3),
     1 TD1(13),TD2(13),TD3(9), TR1(9),TR2(9),TR3(9), U(1),V(1),W(1),
     2 RX(1),RY(1)
      EQUIVALENCE (T11,T1(1)),(T12,T1(2)),(T13,T1(3)),(T21,T2(1)),
     1   (T22,T2(2)),(T23,T2(3)),(T31,T3(1)),(T32,T3(2)),(T33,T3(3)),
     2   (R(1),F(1)),(U(1),FT(1)),(V(1),FT(7)),(W(1),P1(1)),(RX(1),P2(1))
     3   ,(RY(1),P3(1))
      DATA  IPERMQ /2,3,4,1/, MFR /3,3,3,2,2/, WGT /.50,.50,.25/
      LOGICAL QUAD, TRIG, NOMP, NOST, SIST, NOLD, SILD, NOSM, SISM, SKMP
C
C     INITIALIZE
C
      SIST = KKK.LE.0
      NOST = .NOT.SIST
      SILD = KKK.EQ.0.OR.KKK.EQ.1
      NOLD = .NOT.SILD
      SISM = KKK.GE.2
      NOSM = .NOT.SISM
      IF ((NNS.NE.3).AND.(NNS.NE.5))  NNS = 4
      IF ((NPF.NE.3).AND.(NPF.NE.6))  NPF = 5
      NEN = MIN0 (NNS,4)
      QUAD = NEN.EQ.4
      TRIG = NEN.EQ.3
      WG = 1.
      N3 = 2*NEN - 3
      NTRI = 3*NEN - 8
      NEF = NEN*NPF
      IF (MODEX .EQ. 1) RETURN
      NSF = NEF + (NEN-3)*NPF
      IF (MID.NE.4)  MID = 0
      MIDP = MID
      IF (TRIG)  MIDP = 0
      NFM = 3
      IF (NPF.EQ.3)  NFM = 2
      NTF = NSF + NFM*MIDP
      NOMP = MIDP.LE.0
```

```
      SKMP = NOMP.OR.NOST
      IF (NNS.NE.4)  GO TO 130
      X(5) = 0.25*(X(1)+X(2)+X(3)+X(4))
      Y(5) = 0.25*(Y(1)+Y(2)+Y(3)+Y(4))
      Z(5) = 0.25*(Z(1)+Z(2)+Z(3)+Z(4))
      HM(5) = 0.25*(HM(1)+HM(2)+HM(3)+HM(4))
      HP(5) = 0.25*(HP(1)+HP(2)+HP(3)+HP(4))
      IF (KKK.LT.0)  GO TO 130
      T(5) = 0.25*(T(1)+T(2)+T(3)+T(4))
      DT(5) = 0.25*(DT(1)+DT(2)+DT(3)+DT(4))
      DO 110  J = 1,3
      SM(5,J) = 0.25*(SM(1,J)+SM(2,J)+SM(3,J)+SM(4,J))
  110 BM(5,J) = 0.25*(BM(1,J)+BM(2,J)+BM(3,J)+BM(4,J))
      IF (NOLD)  GO TO 130
      P(5) = 0.25*(P(1)+P(2)+P(3)+P(4))
      HW(5) = 0.25*(HW(1)+HW(2)+HW(3)+HW(4))
      DO 120  J = 1,3
  120 RHO(5,J) = 0.25*(RHO(1,J)+RHO(2,J)+RHO(3,J)+RHO(4,J))
  130 IF (NOST)  GO TO 150
      DO 140  I = 1,NTF
      DO 140  J = 1,NTF
  140 S(I,J) =  0.
  150 IF (SISM)  GO TO 170
      DO 160  I = 1,NTF
  160 F(I) = 0.
  170 IF (NOSM.OR.TRIG)  GO TO 200
      NEF1 = NEF + 1
      DO 180  L = NEF1,NTF
      M = L - 1
      DO 180  I = 1,M
  180 R(L) = R(L) - S(I,L)*R(I)
  200 DO 210  I = 1,63
  210 A(I) = 0.
      DO 220  I = 1,3
      CA(I) = CM(1,I)*ALFA(1)+CM(2,I)*ALFA(2)+CM(3,I)*ALFA(3)
      DO 220  J = 1,3
  220 C(I,J) = CM(I,J)
C
C     COMPUTE DIRECTION COSINE MATRIX TO OF LOCAL ELEMENT SYSTEM
C
      CALL QDCOS (NTRI,X,Y,Z,TO)
C
C     LOOP OVER THE NTRI TRIANGLE COMPONENTS
C
      DO 700  NT = 1,NTRI
      N1 = NT
      N2 = IPERMQ(N1)
      NC(1) = N1
      NC(2) = N2
      NC(3) = N3
      MT = MIDP/2
      NOD = 3 + MT
C
C     COMPUTE DIRECTION COSINES OF LOCAL TRIANGLE SYSTEM
C     AND THE TRIANGLE PROJECTIONS A,B ONTO IT
```

```
C
      CALL TDCOS (N1,N2,N3,X,Y,Z,A,B)
C
C     SET UP INPUTS FOR TRIANGLE SUBROUTINES
C
      DO 240  I = 1,3
      L = NC(I)
      LOC(I) = NPF*(L-1)
      HMT(I) = HM(L)
      HPT(I) = HP(L)
      IF (NOLD)  GO TO 240
      ROX = RHO(L,1)
      ROY = RHO(L,2)
      ROZ = RHO(L,3)
      RO1 = T11*ROX+T12*ROY+T13*ROZ
      RO2 = T21*ROX+T22*ROY+T23*ROZ
      RO3 = T31*ROX+T32*ROY+T33*ROZ
      H1 = HW(L)
      P1(I) = RO1*H1
      P2(I) = RO2*H1
      P3(I) = RO3*H1 + P(L)
      TEMP = T(L)
      TMOM = DT(L)*HP(L)**3/12.
      DO 230  J = 1,3
      SMT(I,J) = SM(L,J) - CA(J)*TEMP
  230 BMT(I,J) = BM(L,J) - CA(J)*TMOM
  240 CONTINUE
C
C     FORM TRANSFORMATIONS BETWEEN ELEMENT AND NODAL SYSTEMS
C
      L1 = 9*N1 - 8
      L2 = 9*N2 - 8
      CALL TRFPRD (IDIS,NEN,TDIS(L1),TDIS(L2),TDIS(19),TD1,TD2,TD3)
      IF (NPF.NE.3)
     1CALL TRFPRD (IROT,NEN,TROT(L1),TROT(L2),TROT(19),TR1,TR2,TR3)
      DO 250  I = 7,8
      TD1(I+3) = TD1(I)
      TD1(I+5) = TD1(I)
      TD2(I+3) = TD2(I)
  250 TD2(I+5) = TD2(I)
      LOC(4) = NSF + NFM*(N2-1)
      LOC(5) = NSF + NFM*(N1-1)
      N4 = LOC(4) + 3
      N5 = LOC(5) + 3
C
C     MEMBRANE CONTRIBUTION
C
  260 IF (SISM)  GO TO 320
C     MEMBRANE STIFFNESS AND/OR LOAD VECTOR
      CALL SLST (MT,KKK)
      LT = 0
      DO 300  JJ = 1,NOD
      J = JJ + JJ
      M = LOC(JJ)
      LL = MFR(JJ)
```

```
C
      CALL TDCOS (N1,N2,N3,X,Y,Z,A,B)
C
C     SET UP INPUTS FOR TRIANGLE SUBROUTINES
C
      DO 240  I = 1,3
      L = NC(I)
      LOC(I)  = NPF*(L-1)
      HMT(I)  = HM(L)
      HPT(I)  = HP(L)
      IF (NOLD)  GO TO 240
      ROX = RHO(L,1)
      ROY = RHO(L,2)
      ROZ = RHO(L,3)
      RO1 = T11*ROX+T12*ROY+T13*ROZ
      RO2 = T21*ROX+T22*ROY+T23*ROZ
      RO3 = T31*ROX+T32*ROY+T33*ROZ
      H1 = HW(L)
      P1(I)  = RO1*H1
      P2(I)  = RO2*H1
      P3(I)  = RO3*H1 + P(L)
      TEMP = T(L)
      TMOM = DT(L)*HP(L)**3/12.
      DO 230  J = 1,3
      SMT(I,J)  = SM(L,J)  - CA(J)*TEMP
  230 BMT(I,J)  = BM(L,J)  - CA(J)*TMOM
  240 CONTINUE
C
C     FORM TRANSFORMATIONS BETWEEN ELEMENT AND NODAL SYSTEMS
C
      L1 = 9*N1 - 8
      L2 = 9*N2 - 8
      CALL TRFPRD (IDIS,NEN,TDIS(L1),TDIS(L2),TDIS(19),TD1,TD2,TD3)
      IF (NPF.NE.3)
     1CALL TRFPRD (IROT,NEN,TROT(L1),TROT(L2),TROT(19),TR1,TR2,TR3)
      DO 250  I = 7,8
      TD1(I+3)  = TD1(I)
      TD1(I+5)  = TD1(I)
      TD2(I+3)  = TD2(I)
  250 TD2(I+5)  = TD2(I)
      LOC(4)  = NSF + NFM*(N2-1)
      LOC(5)  = NSF + NFM*(N1-1)
      N4 = LOC(4)  + 3
      N5 = LOC(5)  + 3
C
C     MEMBRANE CONTRIBUTION
C
  260 IF (SISM)  GO TO 320
C     MEMBRANE STIFFNESS AND/OR LOAD VECTOR
      CALL SLST (MT,KKK)
      LT = 0
      DO 300  JJ = 1,NOD
      J = JJ + JJ
      M = LOC(JJ)
      LL = MFR(JJ)
```

```
      DO 300  L = 1,LL
      M = M + 1
      LT = LT + 1
      C1 = TD1(LT)
      C2 = TD2(LT)
      IF (SILD)  F(M) = F(M) + FT(J-1)*C1 + FT(J)*C2
      IF (NOST)  GO TO 300
      KT = 0
      DO 290  II = 1,JJ
      I = II + II
      KK = MFR(II)
      IF (II.EQ.JJ)  KK = L
      H1 = ST(I-1,J-1)*C1 + ST(I-1,J)*C2
      H2 = ST(I  ,J-1)*C1 + ST(I  ,J)*C2
      N = LOC(II)
      DO 290  K = 1,KK
      N = N + 1
      KT = KT + 1
      SQ = S(N,M) + TD1(KT)*H1 + TD2(KT)*H2
      S(N,M) = SQ
  290 S(M,N) = SQ
  300 CONTINUE
      GO TO 400
C     MEMBRANE STRESSES
  320 DO 350 N=1,NOD
      L = LOC(N)
      UE = R(L+1)
      VE = R(L+2)
      IF(N.GT.3) GO TO 330
      WE = R(L+3)
      M3 = 3*N
      M2 = M3-1
      M1 = M2-1
      U(N) = TD1(M1)*UE + TD1(M2)*VE + TD1(M3)*WE
      V(N) = TD2(M1)*UE + TD2(M2)*VE + TD2(M3)*WE
      W(N) = TD3(M1)*UE + TD3(M2)*VE + TD3(M3)*WE
      GO TO 350
  330 U(N) = TD1(7)*UE + TD1(8)*VE
      V(N) = TD2(7)*UE + TD2(8)*VE
  350 CONTINUE
      CALL LSTSTR (MT)
      DO 380 I=1,3
      L = NC(I)
      IF(QUAD) WG = WGT(I)
      TEMP = T(L)
      DO 380 J=1,3
  380 SM(L,J) = SM(L,J) + WG*(SMT(I,J)-CA(J)*TEMP)
  400 IF (NPF.EQ.3)  GO TO 560
C
C     PLATE BENDING CONTRIBUTION
C
      IF (SISM)  GO TO 600
C     BENDING STIFFNESS AND/OR LOAD VECTOR
      CALL SLCCT (MT,KKK)
      DO 500  JJ = 1,3
```

```
      JT = 3*JJ-3
      J = JT + 1
      DO 450  L = 1,NPF
      M = LOC(JJ) + L
      L3 = L - 3
      IF (L3.GT.0)  GO TO 420
      C3 = TD3(JT+L)
      IF (SILD)  F(M) = F(M) + FT(J)*C3
      IF (SKMP)  GO TO 450
      S4 = S(M,N4) + ST(J,10)*C3
      S5 = S(M,N5) - ST(J,11)*C3
      GO TO 430
  420 C1 = TR1(JT+L3)
      C2 = TR2(JT+L3)
      IF (SILD)  F(M) = F(M) + FT(J+1)*C1 + FT(J+2)*C2
      IF (SKMP)  GO TO 450
      S4 = S(M,N4) + ST(J+1,10)*C1 + ST(J+2,10)*C2
      S5 = S(M,N5) - ST(J+1,11)*C1 - ST(J+2,11)*C2
  430 S(M,N4) = S4
      S(N4,M) = S4
      S(M,N5) = S5
      S(N5,M) = S5
  450 CONTINUE
      IF (NOST)  GO TO 500
      DO 480  II = 1,JJ
      IT = 3*II-3
      I = IT + 1
      KK = NPF
      DO 480  L = 1,NPF
      IF (II.EQ.JJ)  KK = L
      M = LOC(JJ) + L
      L3 = L - 3
      IF (L3.GT.0)  GO TO 460
      C3 = TD3(JT+L)
      H1 = ST(I  ,J)*C3
      H2 = ST(I+1,J)*C3
      H3 = ST(I+2,J)*C3
      GO TO 470
  460 C1 = TR1(JT+L3)
      C2 = TR2(JT+L3)
      H1 = ST(I  ,J+1)*C1 + ST(I  ,J+2)*C2
      H2 = ST(I+1,J+1)*C1 + ST(I+1,J+2)*C2
      H3 = ST(I+2,J+1)*C1 + ST(I+2,J+2)*C2
  470 N = LOC(II)
      DO 480  K = 1,KK
      N = N + 1
      K3 = K - 3
      K1 = IT + K
      K2 = IT + K3
      IF (K3.LE.0)  SQ = S(N,M) + TD3(K1)*H1
      IF (K3.GT.0)  SQ = S(N,M) + TR1(K2)*H2 + TR2(K2)*H3
      S(N,M) = SQ
  480 S(M,N) = SQ
  500 CONTINUE
      IF (NOMP)  GO TO 700
```

```
      IF  (NOLD)   GO TO 540
      F(N4)  = F(N4)  + FT(10)
      F(N5)  = F(N5)  - FT(11)
  540 IF  (NOST)   GO TO 700
      S(N4,N4)  = S(N4,N4)  + ST(10,10)
      S(N5,N5)  = S(N5,N5)  + ST(11,11)
      S(N4,N5)  = S(N4,N5)  - ST(10,11)
      S(N5,N4)  = S(N4,N5)
      GO TO 700
  560 IF  (NOLD)   GO TO 700
      FL = (P3(1)+P3(2)+P3(3))*(A(3)*B(2)-A(2)*B(3))/12.
      JT = 0
      DO 580  JJ = 1,2
      DO 580 L=1,3
      JT=JT+1
      M = LOC(JJ) + L
  580 F(M)  = F(M)  + FL*TD3(JT)
      GO TO 700
C     BENDING MOMENTS
  600 DO 650 N=1,3
      L = LOC(N)
      M3 = 3*N
      M2 = M3-1
      M1 = M2-1
      XE = R(L+4)
      YE = R(L+5)
      ZE = 0.0
      IF(NPF.EQ.6)  ZE = R(L+6)
      RX(N)  = TR1(M1)*XE + TR1(M2)*YE + TR1(M3)*ZE
  650 RY(N)  = TR2(M1)*XE + TR2(M2)*YE + TR2(M3)*ZE
      RM(1)  = R(N4)
      RM(2)  =-R(N5)
      CALL LCTMOM  (MT)
      DO 680 I=1,3
      L = NC(I)
      IF(QUAD)  WG = WGT(I)
      TMOM = DT(L)*HP(L)**3/12.0
      DO 680 J=1,3
  680 BM(L,J)  = BM(L,J)  + WG*(BMT(I,J)-CA(J)*TMOM)
  700 CONTINUE
      IF  (SISM.OR.TRIG)   GO TO 900
C
C     CHECK FOR POSSIBLE INTERNAL STIFFNESS SINGULARITY (FLAT
C     OR NEARLY FLAT QUADRILATERAL WHEN NPF = 3 OR 6)
C
      IF  ((NPF.EQ.5).OR.NOST)   GO TO 730
      DO 720  N = 3,6,3
      IF  (NPF.NE.N)   GO TO 720
      M = 5*N
      M1 = M - 1
      M2 = M - 2
      IF  (S(M,M).GT.(S(M1,M1)+S(M2,M2))*1.0E-08)   GO TO 720
      DO 710  I = 1,NTF
      S(I,M) = 0.
  710 S(M,I) = 0.
```

```
  720 CONTINUE
C
C     CONDENSATION OF INTERNAL DEGREES OF FREEDOM
C
  730 NIF = NTF - NEF
      DO 800  N = 1,NIF
      K = NTF - N
      L = K + 1
      PIVOT = S(L,L)
      FL = F(L)
      IF (PIVOT.LE.0.)  GO TO 800
      F(L) = F(L)/PIVOT
      DO 780  I = 1,K
      G = S(I,L)
      IF (G)  740,780,740
  740 IF (NOST)  GO TO 770
      G = G/PIVOT
      S(I,L) = G
      DO 760  J = I,K
      SQ =  S(I,J) - G*S(L,J)
      S(I,J) = SQ
  760 S(J,I) = SQ
  770 F(I) = F(I) - G*FL
  780 CONTINUE
  800 CONTINUE
  900 RETURN
      END
      SUBROUTINE QUAD (B,BB)
C
C     CALLS?  FORMB,VECTOR
C     CALLED BY?  PLNAX
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/    LM(12),S(12,12),P(12,4),XM(12),
     1 TI(20,4),IX(4),IE(5),NS,D(4,4),EMUL(4,5),RR(4),ZZ(4),H(6),HS(6),
     2 HT(6),HR(6),HZ(6),FAC,XMM,PRESS,   EE(10),TTI(4),PP(12,4),THICK
     3 ,TMP(4),TP(12),ALP(4),IFILL2(4236)
      COMMON /JUNK/ MAT,NT,TEMP,REFT,BETA,IFILL1(422)
      DIMENSION B(20,12),BB(20,12)
      DIMENSION SS(2),TT(2),HH(2),SSS(5),TTT(5),IVECT(4),JVECT(4),V(4)
      DATA SSS/0.,-1.,1.,0.,0./, TTT/0.,0.,0.,-1.,1./
      DATA SS/-0.57735026918963,0.57735026918963/
      DATA TT/-0.57735026918963,0.57735026918963/
      DATA HH/1.,1./, IVECT/4,2,1,3/, JVECT/1,3,2,4/
C
      DO 170 J=1,12
      XM(J)=0.0
      TP(J) = 0.0
      DO 160 I=1,20
      BB(I,J)=0.0
  160 B(I,J)=0.0
      DO 170 I=1,12
  170 S(I,J)=0.0
C
      DO 500 II=1,2
```

```
      DO 500 JJ=1,2
      CALL FORMB(SS(II),SS(JJ),B)
      TEMP = 0.0
      DO 200 I=1,4
  200 TEMP = TEMP + H(I)* TMP(I)
      FAC=FAC*HH(JJ)*HH(II)
      FTP = TEMP - REFT
      DO 400 J=1,12
      D1=(D(1,1)*B(1,J)+D(1,2)*B(2,J)+D(1,3)*B(3,J)+D(1,4)*B(4,J))*FAC
      D2=(D(2,1)*B(1,J)+D(2,2)*B(2,J)+D(2,3)*B(3,J)+D(2,4)*B(4,J))*FAC
      D3=(D(3,1)*B(1,J)+D(3,2)*B(2,J)+D(3,3)*B(3,J)+D(3,4)*B(4,J))*FAC
      D4=(D(4,1)*B(1,J)+D(4,2)*B(2,J)+D(4,3)*B(3,J)+D(4,4)*B(4,J))*FAC
      TP(J) = TP(J) + FTP*(D1*ALP(1) +D2*ALP(2) + D3*ALP(3) + D4*ALP(4))
      DO 400 I=J,12
      S(I,J)=S(I,J)+B(1,I)*D1+B(2,I)*D2+B(3,I)*D3+B(4,I)*D4
  400 S(J,I)=S(I,J)
      DO 450 I=1,4
  450 XM(I)=XM(I)+FAC*H(I)
  500 CONTINUE
C
C     FORM STRESS DISDLACEMENT MATRIX
C
      LL=NS/4
      DO 530 L=1,LL
      CALL FORMB(SSS(L),TTT(L),BB)
C
      TEMP = 0.0
      DO 515 K=1,4
  515 TEMP = TEMP + H(K)* TMP(K)
      FAC = TEMP - REFT
      DO 530 II=1,4
      I=II+4*(L-1)
      TI(I,4) = -TTI(II)* FAC
      DO 530 J=1,12
      B(I,J)=0.0
      DO 530 K=1,4
 530  B(I,J)=B(I,J)+D(II,K)*BB(K,J)
C
C     ELIMINATE EXTRA DEGREES OF FREEDOM
C
      IF( IX(3).EQ.IX(4) )  GO TO 560
      IF( NPAR(6).NE.0    )  GO TO 560
      DO 550 NN=1,4
      L=12-NN
      K=L+1
      C = TP(K)/S(K,K)
      DO 535 J=1,NS
  535 TI(J,4) = TI(J,4) + C* B(J,K)
      DO 550 I=1,L
      C=S(I,K)/S(K,K)
      TP(I) = TP(I) - C* TP(K)
      DO 540 J=1,NS
  540 B(J,I)=B(J,I)-C*B(J,K)
      DO 550 J=1,L
  550 S(I,J)=S(I,J)-C*S(K,J)
```

```
C
C     ROTATE STRESS-DISPLACEMENT TRANSFORMATION TO GIVE STRESSES
C     NORMAL AND PARALLEL TO SIDES.  SIMILARLY, ROTATE INITIAL STRESSES.
C
  560 NSET = LL-1
      IF( NSET.LE.0 )  GO TO 730
      DO 720 L=1,NSET
      IV = IVECT(L)
      JV = JVECT(L)
      CALL VECTOR (V,RR(IV),ZZ(IV),0.0,RR(JV),ZZ(JV),0.0)
      S2 = V(1)*V(1)
      C2 = V(2)*V(2)
      SC =-V(1)*V(2)
      I1 = 4*L+1
      I2 = I1+1
      I4 = I1+3
      T1 = TI(I1,4)
      T2 = TI(I2,4)
      T4 = TI(I4,4)
      T5 = 2.0*SC*T4
      TI(I1,4)  = C2*T1+S2*T2+T5
      TI(I2,4)  = S2*T1+C2*T2-T5
      TI(I4,4)  = SC*(T2-T1)+(C2-S2)*T4
      DO 710 J=1,8
      B1 = B(I1,J)
      B2 = B(I2,J)
      B4 = B(I4,J)
      B5 = 2.0*SC*B4
      B(I1,J)  = C2*B1+S2*B2+B5
      B(I2,J)  = S2*B1+C2*B2-B5
  710 B(I4,J)  = SC*(B2-B1)+(C2-S2)*B4
  720 CONTINUE
  730 CONTINUE
C
      DO 660 L=1,4
      DO 600 I=1,NS
  600 TI(I,L)  = TI(I,4)* EMUL(L,1)
      DO 660 I=1,8
  660 P(I,L)  = TP(I)* EMUL(L,1)
C
C     CALCULATE PRESSURE LOADS ON I-J FACE
C
      DR=RR(2)-RR(1)
      DZ=ZZ(1)-ZZ(2)
      RI=PRESS*(2.*RR(1)+RR(2))/6.
      RJ=PRESS*(2.*RR(2)+RR(1))/6.
      IF(NPAR(5).EQ.0) GO TO 670
      RI=PRESS*THICK/2.
      RJ=RI
  670 DO 700 L=1,4
      P(1,L)=P(1,L)+DZ*RI*EMUL(L,2)
      P(5,L)=P(5,L)+DR*RI*EMUL(L,2)
      P(2,L)=P(2,L)+DZ*RJ*EMUL(L,2)
  700 P(6,L)=P(6,L)+DR*RJ*EMUL(L,2)
      RETURN
```

```
      END
       SUBROUTINE REDBAK (A,VA,VV,MAXA,NEQB,NV,NWA,NWV,NWVV,NTB,NBLOCK,
     1MI,MA)
C
C     CALLED BY?  SSPCEB
C
       COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
       DIMENSION A(NWA),VA(NWV),VV(NWVV),MAXA(MI)
C
       INC=NEQB - 1
       NEB=NTB*NEQB
       NEBT=NEB+NEQB
C
C   REDUCE VECTORS ON TAPE NR
       REWIND NRED
       REWIND NR
       REWIND NL
       REWIND NT
       READ (NRED) A,MAXA
       ISV=NTB+1
       IF (NBLOCK.EQ.1) ISV=1
       LL=0
       DO 10 L=1,ISV
       READ (NR) VA
       K=0
       KK=LL
       DO 20 J=1,NV
       DO 30 I=1,NEQB
       K=K+1
       KK=KK+1
 30    VV(KK)=VA(K)
 20    KK=KK+NEB
 10    LL=LL+NEQB
       ISA=1
C
 500   DO 100 N=2,NEQB
       KL=N + INC
       KU=MAXA(N)
       IF (KU-KL) 100,110,110
 110   K=N
       DO 120 L=1,NV
       KJ=K
       DO 130 KK=KL,KU,INC
       KJ=KJ - 1
 130   VV(K)=VV(K) - A(KK)*VV(KJ)
 120   K=K + NEBT
 100   CONTINUE
 135   KL=NEQB
       ML=NEQB + 1
       DO 140 N=ML,MI
       KL=KL + NEQB
       KU=MAXA(N)
       IF (KU-KL) 140,150,150
 150   K=NEQB
       KN=N
```

```
      DO 160 L=1,NV
      KJ=K
      DO 170 KK=KL,KU,INC
      VV(KN)=VV(KN) - A(KK)*VV(KJ)
  170 KJ=KJ - 1
      K=K + NEBT
  160 KN=KN + NEBT
  140 CONTINUE
C
      DO 200 I=1,NEQB
      C=A(I)
      IF (C) 180,200,180
  180 KK=I
      DO 210 L=1,NV
      VV(KK)=VV(KK)/C
  210 KK=KK+NEBT
  200 CONTINUE
      IF (ISA.EQ.NBLOCK) GO TO 400
      READ (NRED) A,MAXA
      ISA=ISA+1
C
C   STORE REDUCED VECTORS ON TAPE NT
      K=0
      KK=0
      DO 240 J=1,NV
      DO 220 I=1,NEQB
      K=K+1
      KK=KK+1
  220 VA(K)=VV(KK)
  240 KK=KK+NEB
      WRITE (NT) VA
      K=1
      DO 310 J=1,NV
      DO 300 I=1,NEB
      VV(K)=VV(K+NEQB)
  300 K=K+1
  310 K=K+NEQB
      IF (ISV.EQ.NBLOCK) GO TO 500
      READ (NR) VA
      ISV=ISV+1
      KK=NEB
      K=0
      DO 330 J=1,NV
      DO 320 I=1,NEQB
      K=K+1
      KK=KK+1
  320 VV(KK)=VA(K)
  330 KK=KK+NEB
      GO TO 500
C
C   BACKSUBSTITUTE VECTORS ON TAPE NT
  400 BACKSPACE NRED
      ISA=1
  420 ML=NEQB+1
      KL=NEQB
```

```
      DO 600 M=ML,MI
      KL=KL+NEQB
      KU=MAXA(M)
      IF (KU-KL) 600,610,610
 610  K=NEQB
      KM=M
      DO 630 L=1,NV
      KJ=K
      DO 620 KK=KL,KU,INC
      VV(KJ)=VV(KJ) - A(KK)*VV(KM)
 620  KJ=KJ - 1
      KM=KM + NEBT
 630  K=K + NEBT
 600  CONTINUE
      N=NEQB
      DO 640 LJ=2,NEQB
      KL=N + INC
      KU=MAXA(N)
      IF (KU-KL) 640,650,650
 650  K=N
      DO 680 L=1,NV
      KJ=K
      DO 690 KK=KL,KU,INC
      KJ=KJ - 1
 690  VV(KJ)=VV(KJ) - A(KK)*VV(K)
 680  K=K + NEBT
 640  N=N - 1
 665  KK=0
      K=0
      DO 660 J=1,NV
      DO 670 I=1,NEQB
      K=K+1
      KK=KK+1
 670  VA(K)=VV(KK)
 660  KK=KK+NEB
      WRITE (NL) VA
      IF (ISA.EQ.NBLOCK) GO TO 800
      BACKSPACE NRED
      READ (NRED) A,MAXA
      BACKSPACE NRED
      ISA=ISA+1
      BACKSPACE NT
      READ (NT) VA
      BACKSPACE NT
      K=NEBT
      DO 700 J=1,NV
      DO 720 I=1,NEB
      VV(K)=VV(K-NEQB)
 720  K=K-1
 700  K=K+NEBT+NEB
      K=0
      KK=0
      DO 740 J=1,NV
      DO 760 I=1,NEQB
      K=K+1
```

```
      KK=KK+1
 760  VV(KK)=VA(K)
 740  KK=KK+NEB
      GO TO 420
 800  RETURN
      END
      SUBROUTINE REDVK (A,VV,MAXA,NEQB,NWA,NEQ,NBLOCK,MI,MA,NCALL)
C
C     CALLED BY?  SOLSTP
C
C     THIS ROUTINE REDUCES AND BACK-SUBSTITUTES A SINGLE VECTOR STORED
C     IN CORE USING A REDUCED MATRIX STORED IN BLOCK FORM.
C
      DIMENSION      A(NWA),VV(NEQ),MAXA(MI)
C
      COMMON /TAPES/ NSTIF,NRED,NL,NR,IFILL(2)
C
      INC=NEQB - 1
      MA1 = MA-1
C
C     PERFORM FORWARD REDUCTION OF THE VECTOR
C
      IF(NBLOCK.EQ.1  .AND.  NCALL.GT.1) GO TO 22
      REWIND NRED
      READ (NRED) A,MAXA
   22 ISA = 1
      KSTART = 2
      KEND = NEQB
C
  500 N = 1
      DO 100 K=KSTART,KEND
      N = N+1
      KL=N + INC
      KU=MAXA(N)
      IF (KU-KL) 100,110,110
  110 KJ = K
      DO 130 KK=KL,KU,INC
      KJ=KJ - 1
  130 VV(K)=VV(K) - A(KK)*VV(KJ)
  100 CONTINUE
C
      IF(ISA.EQ.NBLOCK) GO TO 175
      KL = NEQB
      ML = KEND+1
      MR = MIN0(KEND+MA1,NEQ)
      N = NEQB
      DO 140 K=ML,MR
      N = N+1
      KL=KL + NEQB
      KU=MAXA(N)
      IF (KU-KL) 140,150,150
  150 KJ = KEND
      DO 170 KK=KL,KU,INC
      VV(K) = VV(K) - A(KK)*VV(KJ)
  170 KJ=KJ - 1
```

```
 140   CONTINUE
C
  175 KST = KSTART-1
      N = 0
      DO 200 K=KST,KEND
      N = N+1
      C = A(N)
       IF (C) 180,200,180
  180 VV(K) = VV(K)/C
 200   CONTINUE
  205 IF(ISA.EQ.NBLOCK) GO TO 400
       READ (NRED) A,MAXA
       ISA=ISA+1
      KSTART = KSTART+NEQB
      KEND = MINO(KEND+NEQB,NEQ)
C
       GO TO 500
C
C     BACK-SUBSTITUTE REDUCED VECTOR (STORED IN CORE)
C
  400 IF(ISA.GT.1)
     *BACKSPACE NRED
       ISA=1
      NN = NEQ-(NBLOCK-1)*NEQB
      KEND = NEQ
      GO TO 645
C
  420 KEND = KEND-NN
      NN = NEQB
C
       KL=NEQB
      MR = MINO(NEQ,KEND+MA1)
      ML = KEND+1
      N = NEQB
      DO 600 K=ML,MR
      N = N+1
       KL=KL+NEQB
       KU=MAXA(N)
       IF (KU-KL) 600,610,610
  610 KJ = KEND
       DO 620 KK=KL,KU,INC
       VV(KJ)=VV(KJ) - A(KK)*VV(K)
 620   KJ=KJ - 1
 600   CONTINUE
C
  645 N = NN
      K = KEND
      DO 640 L=2,NN
       KL=N + INC
       KU=MAXA(N)
       IF (KU-KL) 655,650,650
  650  KJ=K
       DO 690 KK=KL,KU,INC
       KJ=KJ - 1
 690   VV(KJ)=VV(KJ) - A(KK)*VV(K)
```

```
  655  N=N - 1
  640 K = K-1
C
      IF (ISA.EQ.NBLOCK) GO TO 800
C
      BACKSPACE NRED
      READ (NRED) A,MAXA
      BACKSPACE NRED
      ISA=ISA+1
C
      GO TO 420
  800 RETURN
      END
      SUBROUTINE RESPEC
      REAL T(4)
C
C     CALLS?  EMIDR,SPECTR,PRINTD,STRESR
C     CALLED BY?  MAIN
C
C
      COMMON /SOL/ NBLOCK,NEQB,LL,NF,IFILL1(7)
      COMMON /JUNK/ XXX(4),NDYN,JUK(421)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
      COMMON /one/ A(1)
C
C
      WRITE (6,1010)
      XXX(4)=0.
C***      CALL TTIME (T(1))
      IF (MODEX.EQ.1) GO TO 100
      N2=N1 + 6*NUMNP
      CALL EMIDR (A(N1),A(N2),NUMNP,NEQB)
C
C
  100 N2=N1+NEQB*NF
      N3=N2+NF*3
      N4=N3+NEQB
      N5=N4+NF
      N6=N5+NEQB
      N7=N6+NF
      MM=N7-MTOT
      IF (MM.GT.0) CALL ERROR (MM)
      CALL SPECTR  (A(N1), A(N2),A(N3),A(N4),A(N5),NEQB,NF,NBLOCK,A(N6))
C
C     MODE SHAPE NG IS R.M.S. DISPLACEMENT
C
C***      CALL TTIME (T(2))
      IF (MODEX.EQ.1) GO TO 200
      N2=N1+6*NUMNP
      NG=NF+1
      N3=N2+6*NG
      N4=N3+NEQB*NG
      MM=N4-MTOT
```

```
      IF (MM.GT.O) CALL ERROR (MM)
      NT=2
      CALL PRINTD(A(N1),A(N2),A(N3),NEQB,NUMNP,NG,NBLOCK,NEQ,NT,3)
C
C     COMPUTE STRESSES
C
C***  200 CALL TTIME (T(3)) !200 IS TRANSFERED TO THE NEXT LINE
200   NSB=NBLOCK*NEQB
      N2=N1+12*NF
      N3=N2+NSB*NF
      N4=N3+12
      MM=N4-MTOT
      IF (MM.GT.O) CALL ERROR(MM)
C
      CALL STRESR (A(N1),A(N2),A(N3),NF,NSB,NEQB,NBLOCK)
C***      CALL TTIME (T(4))
C
      TT=0.
      DO 10 I=1,3
      T(I)=T(I+1)-T(I)
 10   TT=TT+T(I)
      T(4)=TT
      WRITE (6,1000) (T(I),I=1,4)
      RETURN
C
 1000 FORMAT (27H1R. M. S.   T I M E   L O G, /
     1 5X,37HCOMPUTE MAXIMUM NODAL DISPLACEMENTS =, F8.2 /
     2 5X,37HOUTPUT  MAXIMUM NODAL DISPLACEMENTS =, F8.2 /
     3 5X,37HCOMPUTE ELEMENT STRESSES            =, F8.2 //
     4 5X,37HTOTAL FOR SPECTRUM ANALYSIS         =, F8.2 )
 1010 FORMAT (1H1,//34H R E S P O N S E   S P E C T R U M, 3X,
     1              15HA N A L Y S I S, // 1X)
      END
      SUBROUTINE RESPON(W,P,X,NF,NT,NDS)
C
C     CALLED BY?  HISTRY
      REAL KAP
C
      DIMENSION W(NF),P(NT),X(NF,NDS)
      COMMON /DYN/ MT,NOT,XSI,DT,IFILL2(6)
      COMMON /JUNK/ BET,KAP,A(3,3),B(3),U(3),U0(3),IFILL1(390)
C
C     EVALUATION OF NORMAL RESPONSE
C
      REWIND 7
      REWIND 4
      READ (7) W
      TH=1.4
C
      DO 260 N=1,NF
      READ (4) P
      K=1
      NOUT=NOT+1
      BET = 1. / (TH/(W(N)*W(N)*DT*DT) + XSI*TH*TH/(W(N)*DT) + TH*TH*TH/
     16 )
```

```
      KAP=XSI*BET/(W(N)*DT)
      A(1,1)=1. - BET*TH*TH/3. - 1./TH - KAP*TH
      A(2,1)=DT*(1. - 1./(2.*TH) - BET*TH*TH/6. - KAP*TH/2.)
      A(3,1)=DT*DT*(0.5 - 1./(6.*TH) - BET*TH*TH/18. - KAP*TH/6.)
      A(1,2)=(-BET*TH - 2.*KAP)/DT
      A(2,2)=1. - BET*TH/2. - KAP
      A(3,2)=DT*(1. - BET*TH/6. - KAP/3.)
      A(1,3)=-BET/(DT*DT)
      A(2,3)=-BET/(2.*DT)
      A(3,3)=1. - BET/6.
      B(1)=BET/(W(N)*W(N)*DT*DT)
      B(2)=BET/(2.*W(N)*W(N)*DT)
      B(3)=BET/(6.*W(N)*W(N))
      DO 230 J=1,3
      U0(J)=0.
  230 U(J)=0.
      U0(1)=P(1)
C
      DO 260 I=2,NT
      DO 240 L=1,3
      U(L)=B(L)*P(I)
      DO 240 LL=1,3
  240 U(L)=U(L) + A(L,LL)*U0(LL)
      DO 245 L=1,3
  245 U0(L)=U(L)
      IF(NOUT-I) 260,250,260
  250 X(N,K)=U(3)
      K=K+1
      NOUT=NOUT+NOT
  260 CONTINUE
C
      REWIND 4
      WRITE (4) X
C
      RETURN
C
      END
      SUBROUTINE SBLOCK(VOLD,VNEW,XM,NFO,NV,NEQBO,NEQB,NBLOKO,NBLOCK)
C
C     CALLED BY?  MODES
C
      COMMON /TAPES/ NSTIF,NRED,NL,NR,NT,NMASS
      DIMENSION VOLD(NEQBO,NFO),VNEW(NEQB,NV),XM(NEQB)
      READ (10) (VOLD(1,I),I=1,NFO)
      DO 260 L=1,NBLOKO
      READ (10) VOLD
  260 CONTINUE
      LBLOKO=1
      LBLOCK=0
C
      I=0
      K=0
      REWIND NMASS
      READ (NMASS) XM
C
```

```
      REWIND NT
      BACKSPACE 10
C
      GO TO 240
C
  200 K=K+1
      I=I+1
      XMM=XM(I)
      DO 100 J=1,NFO
  100 VNEW(I,J)=VOLD(K,J)*XMM
C
      IF (K.LT.NEQBO) GO TO 120
      K=0
      LBLOKO=LBLOKO+1
      IF (LBLOKO-NBLOKO) 140,140,160
C
C
  140 BACKSPACE 10
      READ (10) VOLD
      BACKSPACE 10
C
  120 IF (I.LT.NEQB) GO TO 200
      I=0
C
  160 LBLOCK=LBLOCK+1
C
      WRITE (NT) VNEW
C
      IF (LBLOCK.EQ.NBLOCK) RETURN
      READ (NMASS) XM
  240 DO 220 LI=1,NEQB
      DO 220 LJ=1,NV
  220 VNEW(LI,LJ)=0.0
C
      GO TO 200
C
      END
      SUBROUTINE SCHECK (DL,RTOLV,A,XM,BUP,BLO,BUPC,NEIV,NWA,NEQB,
     1NBLOCK,NF,NV,SHIFT,NEI,IFPR,RTOL)
C
C     CALLED BY?  SSPCEB
C
      COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
      DIMENSION A(NWA),XM(NEQB),BUP(NV),BLO(NV),BUPC(NV),DL(NV),
     1RTOLV(NV)
      INTEGER NEIV(NV)
C
      FTOL=1.0E-02
C
      DO 100 I=1,NV
      BUP(I)=DL(I)*(1.0+FTOL)
  100 BLO(I)=DL(I)*(1.0-FTOL)
      NROOT=0
      DO 120 I=1,NF
  120 IF (RTOLV(I).LT.RTOL) NROOT=NROOT+1
```

```
      IF (NROOT.GE.1) GO TO 200
      WRITE (6,1010)
      STOP
C
C     FIND UPPER BOUNDS ON EIGENVALUE CLUSTERS
  200 DO 240 I=1,NROOT
  240 NEIV(I)=1
      IF (NROOT.NE.1) GO TO 260
      BUPC(1)=BUP(1)
      LM=1
      L=1
      I=2
      GO TO 295
  260 L=1
      I=2
  270 IF (BUP(I-1).LE.BLO(I)) GO TO 280
      NEIV(L)=NEIV(L)+1
      I=I+1
      IF (I.LE.NROOT) GO TO 270
  280 BUPC(L)=BUP(I-1)
      IF (I.GT.NROOT) GO TO 290
      L=L+1
      I=I+1
      IF (I.LE.NROOT) GO TO 270
      BUPC(L)=BUP(I-1)
  290 LM=L
  295 IF (BUP(I-1).LE.BLO(I)) GO TO 300
      IF (RTOLV(I).GT.RTOL) GO TO 300
      BUPC(L)=BUP(I)
      NEIV(L)=NEIV(L)+1
      NROOT=NROOT+1
      IF (NROOT.EQ.NV) GO TO 300
      I=I+1
      GO TO 295
C
C     FIND SHIFT
  300 WRITE (6,1020)
      WRITE (6,1005) (BUPC(I),I=1,LM)
      WRITE (6,1030)
      WRITE (6,1006) (NEIV(I),I=1,LM)
      LL=LM-1
      IF (LM.EQ.1) GO TO 310
  330 DO 320 I=1,LL
  320 NEIV(L)=NEIV(L)+NEIV(I)
      L=L-1
      LL=LL-1
      IF (L.NE.1) GO TO 330
  310 WRITE (6,1040)
      WRITE (6,1006) (NEIV(I),I=1,LM)
      L=0
      DO 340 I=1,LM
      L = L+ 1
      IF (NEIV(I).GE.NROOT) GO TO 350
  340 CONTINUE
  350 SHIFT=BUPC(L)
```

```
      NEI=NEIV(L)
C
C     SHIFT MATRIX
      REWIND NSTIF
      REWIND NMASS
      REWIND NRED
      DO 400 L=1,NBLOCK
      READ (NSTIF) A
      READ (NMASS) XM
      DO 420 I=1,NEQB
 420  A(I)=A(I)-SHIFT*XM(I)
      WRITE (NRED) A
 400  CONTINUE
      I=NSTIF
      NSTIF=NRED
      NRED=I
      RETURN
C
 1005 FORMAT (1H0,6E22.14)
 1006 FORMAT (1H0,6I22)
 1010 FORMAT (37H0***ERROR   SOLUTION STOP IN *SCHECK*, / 12X,
     1        21HNO EIGENVALUES FOUND., / 1X)
 1020 FORMAT (37H0UPPER BOUNDS ON EIGENVALUE CLUSTERS  )
 1030 FORMAT (34H0NO OF EIGENVALUES IN EACH CLUSTER  )
 1040 FORMAT (42H0NO OF EIGENVALUES LESS THAN UPPER BOUNDS  )
      END
      FUNCTION   SD(TT)
C
C     CALLED BY?  SPECTR
C
      COMMON / JUNK / MM,L,K,NTAG,NDYN,I,T(90),S(90),HED(12),W,SS,SI,
     1                TI,IFILL1(32)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
      IF (NTAG.EQ.1) GO TO 500
      NTAG=1
C
C     READ SPECTRUM (MAX DISPL AS FUNCTION OF PERIOD)
C
      READ (5,1000) HED
      WRITE (6,2000) HED
      READ (5,1010) NPTS,SFTR
      IF (ABS(SFTR) .LT. 1.0D-12) SFTR=1.0D0
      WRITE (6,2010) NPTS,SFTR
      READ (5,1020) (T(I),S(I),I=1,NPTS)
      WRITE (6,2020) (I,T(I),S(I),I=1,NPTS)
      IF (MODEX.EQ.1) RETURN
  500 CONTINUE
C
      K=0
      DO 600 I=1,NPTS
      K=K+1
      IF (TT.LT.T(I)) GO TO 700
  600 CONTINUE
  700 TK=T(K)-T(K-1)
      SK=S(K)-S(K-1)
```

```
      SS=S(K-1) + SK*(TT-T(K-1))/TK
      SD=SFTR*SS
C
 1000 FORMAT (12A6)
 1010 FORMAT (I5,F10.0)
 1020 FORMAT (2F10.0)
C
 2000 FORMAT (//17H SPECTRUM TABLE (,12A6,1H),/ 1X)
 2010 FORMAT (5X,18HNUMBER OF POINTS =, I4/
     1         5X,18HSCALE FACTOR      = E14.5 / 1X )
 2020 FORMAT (6H INPUT,20X,8HSPECTRUM, / 6H POINT,8X,6HPERIOD,9X,
     1 5HVALUE, / (I6,2E14.4) )
C
      RETURN
      END
      SUBROUTINE SDSPLY (TEMP,X,MMX,MAX,NCL,NUM,NN,KKK,ISD,ISP,NPT,KT)
C
C     CALLS?  SPLOT,ELOUTS
C     CALLED BY?  STEP
C
C     SUBROUTINE TO PRINT RESPONSE TABLES, TO PRODUCE PRINTER PLOTS
C     OF DISPLACEMENT OR STRESS COMPONENTS, OR TO RECOVER MAXIMA ONLY
C
C     ISD = 1, STRESSES          KKK = 1, PRINT RESPONSE TABLES + MAXIMA
C     ISD = 2, DISPLACEMENTS     KKK = 2, PRINTER PLOTS         + MAXIMA
C                                KKK = 3, RECOVER                 MAXIMA
C
      COMMON /ELPAR/ NPAR(14),IFILL1(10)
      COMMON /EM/ SSA(8,63),KLM(8,63)
      COMMON /JUNK/  NDUM(6),NBL,LAST,KD(2,8),TM(8),DM(8),D(8),IFLL(358)
      COMMON /DYN/   NT,NOT,DAMP,DT,BETA,IFILL4(4)
C
      DIMENSION      TEMP(MAX,NCL),X(MMX,NCL),NUM(NN)
C
C     SET TAPE ASSIGNMENTS
C
C        1. FILE FOR COMPACTING *TEMP* RECORDS INTO *X* RECORDS
C
      IT = 3
C
C        2. FILE *KT* IS *TAPE4* IF DISPLACEMENTS ARE TO BE OUTPUT
C           FILE *KT* IS *TAPE7* IF STRESSES      ARE TO BE OUTPUT
C
      REWIND KT
C
C        3. *TAPE9* CONTAINS OUTPUT REQUESTS AND ELEMENT CONTROL DATA
C
      NT9 = 9
C
C        4. *TAPE8* CONTAINS ELEMENT STRESS/DISPLACEMENT TRANSFORMATION
C           MATRICES PACKED AS 8 COMPONENTS PER RECORD
C
      NT8 = 8
      REWIND NT8
C
```

```
C         5. SCRATCH FILES FOR PRODUCTION OF PRINTER PLOTS
C
      NT2 = 1
      NT4 = 2
C
C     IF *X* IS LARGER THAN *TEMP*, PACK *TEMP* RECORDS INTO *X* --
C     OTHERWISE PASS
C
      IF(MAX.NE.MMX)  GO TO 25
      IT=KT
      NBLOCK = NBL
      GO TO 80
C
C     STORE *NBL* RECORDS OF *TEMP* IN THE LARGER ARRAY *X* (I.E.,
C     MMX.GT.MAX)
C
   25 K=0
      REWIND IT
      NBLOCK = 0
      DO 75 NB=1,NBL
      READ (KT) TEMP
      DO 50 I=1,MAX
      II=I+K
      DO 50 J=1,NCL
   50 X(II,J)=TEMP(I,J)
      K=K+MAX
      L = K+MAX
      IF(L.LE.MMX) GO TO 75
      WRITE (IT) X
      K=0
      NBLOCK=NBLOCK+1
   75 CONTINUE
C
      IF(K.EQ.0) GO TO 80
      WRITE (IT) X
      NBLOCK = NBLOCK +1
C
   80 IF=0
C                 /NN=1      , FOR DISPLACEMENT OUTPUT/
C                 /NN=NELTYP, FOR STRESS       OUTPUT/
      DO 900 N=1,NN
C
      REWIND NT2
      REWIND NT4
C         SET THE NUMBER OF OUTPUT RECORDS TO BE PROCESSED FROM *TAPE9*
      MM=NUM(N)
C         READ ELEMENT CONTROL PARAMETERS FOR STRESS OUTPUT
      IF(ISD.EQ.2) GO TO 90
      READ (NT9) NPAR
      MTYPE=NPAR(1)
   90 IF(MM.EQ.0) GO TO 900
C
C     LOOP ON THE TOTAL NUMBER OF OUTPUT RECORDS ON *TAPE9*
C
      DO 600 M=1,MM
```

```
C
      REWIND IT
      IF(ISD.EQ.1) READ (NT8) ND,((SSA(I,J),I=1,8),J=1,ND),
     1                            ((KLM(I,J),I=1,8),J=1,ND)
                   READ (NT9) KD,L
C
      GO TO (100,300,200),KKK
C
C     LABEL HEADINGS FOR PRINTED TIME HISTORY OUTPUT
C
  100 IF(ISD.EQ.1) GO TO 130
      WRITE (6,1000) M
      WRITE (6,2001) (KD(1,I),KD(2,I),I=1,L)
      GO TO 300
  130 CALL ELOUTS (KD,L,MTYPE,M,ND)
      GO TO 300
C
C     LABEL HEADINGS FOR PRINTING OF MAXIMA
C
  200 IF(M.GT.1) GO TO 300
      IF(ISD.EQ.1) GO TO 230
      WRITE (6,1002)
      WRITE (6,5001)
      GO TO 300
  230 WRITE (6,2002) MTYPE
      WRITE (6,4001)
C
C     COMPUTE HISTORY
C
  300 DO 320 I=1,L
      TM(I)=0.
  320 DM(I)=0.
      TIME=0.
C
C     READ DISPLACEMENT HISTORY IN BLOCKS
C
      NR = MMX
C
      DO 505 NB=1,NBLOCK
C
      READ (IT) X
C         PROCESS *NR* OUTPUT STEPS IN THIS BLOCK
      IF(NB.LT.NBLOCK) GO TO 325
      NR = NPT - (NBLOCK-1)*MMX
  325 CONTINUE
      DO 500 K=1,NR
      TIME=TIME + DT
      DO 450 I=1,L
      GO TO (330,360),ISD
C         COMPUTE STRESSES
  330 DD=0.
      DO 350 J=1,ND
      JJ=KLM(I,J)
      IF(JJ) 350,350,340
  340 DD=DD+SSA(I,J)*X(K,JJ)
```

```
  350 CONTINUE
      GO TO 400
C         SELECT THE DISPLACEMENT COMPONENT
  360 JJ = IF+1
      DD = X(K,JJ)
C         UPDATE THE MAXIMUM VALUE OF THE COMPONENT
  400 AD=ABS(DD)
      IF(AD-DM(I)) 450,450,445
  445 DM(I)=AD
      TM(I)=TIME
  450 D(I)=DD
C
      GO TO (480,490,500),KKK
C
C     PRINT HISTORY OUTPUT
C
  480 WRITE (6,1004) TIME,(D(I),I=1,L)
      GO TO 500
C
C     SAVE DISPLACEMENTS FOR THE PRODUCTION OF PLOTS
C
  490 WRITE (NT4) D
C
  500 CONTINUE
C
  505 CONTINUE
C
C     COMPLETE THIS OUTPUT SET
C
      GO TO (510,520,530),KKK
C         MAXIMA AT THE END OF A PRINTED HISTORY
  510 WRITE (6,1005) (DM(I),I=1,L)
      WRITE (6,1006) (TM(I),I=1,L)
      GO TO 600
C         SAVE OUTPUT SET DATA FOR PRINTER PLOTS
  520 WRITE (NT2) KD,DM,TM,L
      GO TO 600
C         PRINT SUMMARY OF MAXIMA ONLY
  530 WRITE (6,1007) (KD(1,I),KD(2,I),DM(I),TM(I),I=1,L)
C
  600 IF=IF+L
C
C     PLOT SET OF VALUES
C
      IF(KKK.NE.2) GO TO 900
      REWIND NT2
      REWIND NT4
C
      DO 800 M=1,MM
      GO TO (610,620),ISD
C
  610 WRITE (6,4000) MTYPE,M
      WRITE (6,4001)
      GO TO 630
C
```

```
  620 WRITE (6,5000) M
      WRITE (6,5001)
C
  630 CALL SPLOT (NT2,NT4,NPT,ISP)
C
  800 CONTINUE
C
  900 CONTINUE
C
      RETURN
C
C     F O R M A T S
C
 1000 FORMAT (50H1D I S P L A C E M E N T   T I M E   H I S T O R Y, //
     1 13H OUTPUT SET =,I4, // 14X,27H*NODE NUMBER* - (COMPONENT ,
     2  7HNUMBER), 1X)
 1002 FORMAT (38H1D I S P L A C E M E N T  M A X I M A, // 1X)
 1004 FORMAT (F12.5,2X,1P8E12.3)
 1005 FORMAT (/ 24H MAXIMUM ABSOLUTE VALUES, // 8H MAXIMUM,6X,1P8E12.3)
 1006 FORMAT (5H TIME,9X,1P8E12.3)
 1007 FORMAT (I8,12X,I3,1P2E14.4,7X,2HNA)
 2001 FORMAT (8X,4HTIME,2X, 8(3X,I4,2H-(,I2,1H)) / 1X)
 2002 FORMAT (46H1S T R E S S   C O M P O N E N T   M A X I M A, //
     1        22H ELEMENT TYPE NUMBER =, I3, // 1X)
 4000 FORMAT (51H1N O R M A L I Z E D   S T R E S S   H I S T O R Y,3X,
     1  7HP L O T, // 22H ELEMENT TYPE NUMBER =, I3 /
     2               22H OUTPUT SET NUMBER   =, I3 // 1X)
 4001 FORMAT (8H ELEMENT,9X,6HSTRESS,7X,7HMAXIMUM,7X,7HTIME AT,5X,
     1 4HPLOT,/ 8H  NUMBER,6X,9HCOMPONENT,9X,5HVALUE,7X,7HMAXIMUM,3X,
     2 6HSYMBOL, / 1X)
 5000 FORMAT (46H1N O R M A L I Z E D   D I S P L A C E M E N T,3X,
     1 23HH I S T O R Y   P L O T, // 22H OUTPUT SET NUMBER   =, I3//1X)
 5001 FORMAT (4X,4HNODE,3X,12HDISPLACEMENT,7X,7HMAXIMUM,7X,7HTIME AT,
     2 5X,4HPLOT, / 8H  NUMBER,6X,9HCOMPONENT,9X,5HVALUE,7X,7HMAXIMUM,
     3 3X,6HSYMBOL, / 1X)
C
      END
      SUBROUTINE  SECNTD (A,B,V,MAXA,W,VV,WW,ROOT,TIM,ERRVL,ERRVR,
     1NITE,N,MA,NROOT,NC,IFPR,ANORM,COFQ)
      REAL TIM1,TIM2,TIM3
C
C     CALLS?  BANDET
C     CALLED BY?  MODES
C
      COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
      DIMENSION A(N,NC),B(N),V(1),W(1),VV(N,1),WW(N,1),ROOT(1),
     1TIM(1),ERRVL(1),ERRVR(1)
      INTEGER NITE(1),MAXA(1)
      COMMON /EM/ AT(1000),IFILL(3138)
C
C     THE FOLLOWING TOLERANCES ARE SET FOR THE IBM 370
      ACTOL=1.0D-04
      RCBTOL=1.D-05
      RTOL=1.0D-10
      RQTOL=1.0D-12
```

```
C     SCALE=2.0D0**200
C
       NTF=5
       IITEM=10
       NITEM=60
       NVM=6
C
       REWIND NT
       REWIND NMASS
       READ (NMASS) B
C
C      ETA=2.0
       NOV=0
       JR=1
       NSK=0
       NWA=N*MA
       ISC=1000
C
C   FIND LOCATIONS FOR NEGATIVE ELEMENTS IN STARTING ITERATION VECTORS
C
      REWIND NSTIF
      READ (NSTIF) (A(I,1),I=1,N)
      DO 1 I=1,N
      AA=A(I,1)
      IF (AA.GT.0.) GO TO 1
      WRITE (6,1000) I,AA
      STOP
 1    V(I)=B(I)/AA
      DO 2 J=3,NC
      RMAX=0.
      DO 3 I=1,N
      IF (V(I).LT.RMAX) GO TO 3
      RMAX=V(I)
      IMAX=I
 3    CONTINUE
      NITE(J)=IMAX
 2    V(IMAX)=0.
C
C  CHECK FOR SINGLE DEGREE-OF-FREEDOM SYSTEM
C
      IF (N.GT.1) GO TO 5
      IF(B(1).GT.0.) GO TO 7
      WRITE(6,1180)
      STOP
    7 REWIND NSTIF
      READ(NSTIF) A(1,1)
      ROOT(1)=A(1,1)/B(1)
      NSCH=1
      A(1,1)=1.0D0/SQRT(B(1))
      GO TO 950
C
C***    5 CALL TTIME(TIM1) !5 IS TRANSFERED TO THE NEXT LINE
5       RA=0.0
        RR=0.0
       KA=0
```

```
      KB=0
      KR=0
      CALL BANDET (A,B,V,MAXA,N,NWA,RA,NSCH,DETA,ISC,1)
      FA=DETA
      FR=FA
      DETR=DETA
C
C   CHECK FOR ZERO EIGENVALUE(S)
      IF (A(N,1) .GT. ANORM)  GO TO 10
      WRITE (6,1009)
      STOP
C
C   FIND LOWER BOUND ON SMALLEST EIGENVALUE
 10   IF (IFPR.EQ.1)
     * WRITE(6,1010)
      DO 100 I=1,N
 100  W(I)=B(I)
      RT=0.0
      IITE=0
      KK=2
 110  IITE=IITE+1
      DO 120 I=1,N
 120  V(I)=W(I)
      CALL BANDET (A,B,V,MAXA,N,NWA,RA,NSCH,DETA,ISC,KK)
      KK=2
      RQT=0.0
      DO 130 I=1,N
 130  RQT=RQT+W(I)*V(I)
      DO 180 I=1,N
 180  W(I)=B(I)*V(I)
      RQB=0.0
      DO 140 I=1,N
 140  RQB=RQB+W(I)*V(I)
      RQ=RQT/RQB
      IF (IFPR.EQ.1)
     * WRITE (6,1004) RQ
      BS=SQRT(RQB)
      TOL=ABS(RQ-RT)/RQ
      IF (TOL.LT.RCBTOL) GO TO 150
      DO 160 I=1,N
 160  W(I)=W(I)/BS
      RT=RQ
      IF (IITE.LT.IITEM) GO TO 110
C
 150  DO 170 I=1,N
 170  V(I)=V(I)/BS
      RB=RQ*(1.0D0-DMIN1(1.0D-1,1.0D2*TOL))
      IS=0
 230  CALL BANDET (A,B,V,MAXA,N,NWA,RB,NSCH,DETB,ISC,1)
      IF (IFPR.EQ.1)
     * WRITE (6,1020) RB,NSCH
      FB=DETB
C     IF (NSCH.EQ.0) GO TO 300
      IF (NSCH.EQ.0) GO TO 299
      IS=IS+1
```

```
      IF (IS.LE.NTF) GO TO 240
      WRITE (6,1030) NTF
      STOP
 240  RB=RB/(NSCH+1)
      GO TO 230
C
C   ITERN FOR INDIVIDUAL ROOTS
  299 ETA=2.D0
 300  IF (IFPR.EQ.1)
     * WRITE (6,1040)
      NITE(JR)=-1
      IF (IFPR.EQ.1)
     * WRITE (6,1050) JR,NITE(JR),RA,DETA,FA,ETA,ISC
      NITE(JR)=0
      IF (IFPR.EQ.1)
     * WRITE (6,1050) JR,NITE(JR),RB,DETB,FB,ETA,ISC
C
C   WE STOP WHEN WE HAVE THE REQUIRED NO OF ROOTS SMALLER THAN RC AND
C   NOV=0
 310  IF (NSCH.GE.NROOT) GO TO 900
      IF (RB.GT.COFQ) GO TO 900
C
      IF (KB-KA) 301,303,302
C***  301 FB=FB*1.D50
  301 FB=FB*1.D38
      KB=KB+1
      GO TO 303
C***  302 FA=FA*1.D50
  302 FA=FA*1.D38
      KA=KA+1
C      DIF=FB-FA
  303 DIF=FB-FA
      IF (DIF.NE.0.0) GO TO 320
      WRITE (6,1060)
      GO TO 900
 320  DEL=FB*(RB-RA)/DIF
      RC=RB-ETA*DEL
      TOL=RCBTOL*RC
      IF (ABS(RC-RB) .GT. TOL) GO TO 330
      IF (IFPR.EQ.1)
     * WRITE (6,1070)
      ROOT(JR)=RB
      GO TO 400
C
 330  CALL BANDET (A,B,V,MAXA,N,NWA,RC,NSCH,DETC,ISC,1)
      FC=DETC
      KC=0
      NITE(JR)=NITE(JR)+1
      IF (JR.EQ.1) GO TO 340
      JJ=JR-1
      DO 350 K=1,JJ
C***      IF (ABS(FC).GT.1.D-50) GO TO 350
      IF (ABS(FC).GT.1.D-38) GO TO 350
C***     FC=FC*1.D50
      FC=FC*1.D38
```

```
      KC=KC+1
 350  FC=FC/(RC-ROOT(K))
 340  IF (IFPR.EQ.1)
     * WRITE (6,1050) JR,NITE(JR),RC,DETC,FC,ETA,ISC
C
C   IF WE HAVE MORE SIGNCHANGES THAN EIGENVALUES SMALLER THAN RC WE
C   START INV. ITERATION
      NES=0
      IF (JR.EQ.1) GO TO 380
      DO 360 I=1,JJ
 360  IF (ROOT(I).LT.RC) NES=NES+1
 380  NOV=NSCH-NES
      IF (NOV.EQ.0) GO TO 370
      IF (IFPR.EQ.1)
     * WRITE (6,1080) NOV
      ROOT(JR)=RC
      IF (NOV.GT.1) NSK=1
C
      GO TO 400
 370  RR=RA
      FR=FA
      DETR=DETA
      RA=RB
      FA=FB
      DETA=DETB
      RB=RC
      FB=FC
      DETB=DETC
      KR=KA
      KA=KB
      KB=KC
C
C   WE RESET ETA IF NECESSARY
      TOL=RB*ACTOL
      IF (ABS(RA-RB) .LT. TOL) ETA=ETA*2.0D0
      IF (NITE(JR).LE.NITEM) GO TO 310
      WRITE (6,1015) JR,NITE(JR)
      GO TO 900
C
C   CHECK FOR STORAGE
 400  IF (JR.LE.NC) GO TO 405
      WRITE (6,1090)
      GO TO 900
C
 405  NOR=JR-1
      IF (NOR.GT.NVM) NOR=NVM
C***      CALL TTIME(TIM3)
      IF (IFPR.EQ.1)
     * WRITE (6,1100) NOR
      IF (JR.EQ.1) GO TO 410
      DO 420 I=1,N
 420  V(I)=1.0
      KK=2
      IF (JR.EQ.NC) GO TO 410
      I = NITE(JR+1)
```

```
      V(I) = -1.
 410  DO 430 I=1,N
 430  W(I)=B(I)*V(I)
      IS=0
      GO TO 510
C
C   INVERSE ITERN
 440  NITE(JR)=NITE(JR)+1
      DO 450 I=1,N
 450  V(I)=W(I)
      CALL BANDET (A,B,V,MAXA,N,NWA,RC,NSCH,DETC,ISC,KK)
      IF (IS.EQ.1) GO TO 460
      KK=2
      RQT=0.0
      DO 470 I=1,N
 470  RQT=RQT+W(I)*V(I)
      DO 475 I=1,N
 475  W(I)=B(I)*V(I)
      RQB=0.0
      DO 480 I=1,N
 480  RQB=RQB+W(I)*V(I)
      RQ=RQT/RQB
      RT=ROOT(JR)+RQ
      IF (IFPR.EQ.1)
     * WRITE (6,1110) JR,NITE(JR),RT,RQ
      TOL=RT*RQTOL
      IF (ABS(RT-RTA) .GT. TOL)  GO TO 510
      IS=1
      GO TO 440
C
 510  RTA=RT
      BS=SQRT(RQB)
      DO 490 I=1,N
 490  W(I)=W(I)/BS
      IF (NOR.EQ.0) GO TO 550
      DO 520 K=1,NOR
      AL=0.0
      DO 530 I=1,N
 530  AL=AL+VV(I,K)*W(I)
      DO 540 I=1,N
 540  W(I)=W(I)-AL*WW(I,K)
 520  CONTINUE
C
 550  IF (NITE(JR).LE.NITEM) GO TO 440
      WRITE (6,1015) JR,NITE(JR)
      GO TO 900
C
 460  RQT=0.0
      ERRT=RQB
      DO 570 I=1,N
 570  RQT=RQT+V(I)*W(I)
      DO 560 I=1,N
 560  W(I)=B(I)*V(I)
      RQB=0.0
      DO 580 I=1,N
```

```
 580    RQB=RQB+V(I)*W(I)
C
C   OBTAIN A RATHER LARGE ERROR BOUND
        RQ=RQT/RQB
        ROOT(JR)=ROOT(JR)+RQ
        ERR=SQRT(ERRT/RQB)
        ERRVL(JR)=ROOT(JR)-ERR
        ERRVR(JR)=ROOT(JR)+ERR
C
        BS=SQRT(RQB)
        DO 590 I=1,N
        W(I)=W(I)/BS
 590    V(I)=V(I)/BS
        JJ=JR
        IF (JJ.LE.NVM) GO TO 610
        WRITE (NT) (VV(J,1),J=1,N)
        DO 600 K=1,N
        DO 600 L=2,NVM
        WW(K,L-1)=WW(K,L)
 600    VV(K,L-1)=VV(K,L)
        JJ=NVM
 610    DO 620 K=1,N
        WW(K,JJ)=W(K)
 620    VV(K,JJ)=V(K)
C
C***      CALL TTIME(TIM2)
        TIM3=TIM2-TIM3
        IF (IFPR.EQ.1)
     *  WRITE (6,1120) TIM3
        TIM(JR)=TIM2-TIM1
        TIM1=TIM2
C
C   DECIDE STRATEGY FOR ITERN TOWARDS NEXT ROOT
        TOL=RTOL*ROOT(JR)
        IF (NOV.GT.0) GO TO 700
        IF (ABS(ROOT(JR)-RB) .GT. TOL) GO TO 710
        IF (RA.GT.0.0) GO TO 720
        RA=RB/2.
        CALL BANDET (A,B,V,MAXA,N,NWA,RA,NSCH,DETA,ISC,1)
        FA=DETA
        KA=0
 720    RB=RA
        FB=FA
        KB=KA
        KA=KR
        DETB=DETA
        RA=RR
        FA=FR
        DETA=DETR
        GO TO 710
C
 700    IF (ROOT(JR).GT.RC) NSK=1
        IF (NSK.EQ.1) GO TO 730
        IF (ABS(RC-ROOT(JR)) .LT. TOL) GO TO 740
        IF (ABS(ROOT(JR)-RB) .LT. TOL) GO TO 750
```

```
      RA=RB
      FA=FB
      DETA=DETB
     KA=KB
 750  RB=RC
      FB=FC
     KB=KC
      DETB=DETC
      GO TO 710
  740 IF (ABS(ROOT(JR)-RB) .GT. TOL) GO TO 710
      IF (RA.GT.0.0) GO TO 760
      RA=RB/2.
      CALL BANDET (A,B,V,MAXA,N,NWA,RA,NSCH,DETA,ISC,1)
      FA=DETA
     KA=0
 760  RB=RA
      FB=FA
     KB=KA
     KA=KR
      DETB=DETA
      RA=RR
      FA=FR
      DETA=DETR
 710  FA=FA/(RA-ROOT(JR))
      FB=FB/(RB-ROOT(JR))
      JR=JR+1
C***     IF (ABS(FA).GT.1.D-50) GO TO 711
     IF (ABS(FA).GT.1.D-38) GO TO 711
C***     FA=FA*1.D50
     FA=FA*1.D38
     KA=KA+1
C*** 711 IF (ABS(FB).GT.1.D-50) GO TO 299
  711 IF (ABS(FB).GT.1.D-38) GO TO 299
C***     FB=FB*1.D50
     FB=FB*1.D38
     KB=KB+1
C     ETA=2.0
C     GO TO 300
     GO TO 299
C
 730  IF (RA.GT.0.0) GO TO 780
      RA=RB/2.
      CALL BANDET (A,B,V,MAXA,N,NWA,RA,NSCH,DETA,ISC,1)
      FA=DETA
     KA=0
 780 IF (ABS(ROOT(JR)-RB).GT.TOL) GO TO 770
      RB=RA
      FB=FA
     KB=KA
     KA=KR
      DETB=DETA
      RA=RR
      FA=FR
      DETA=DETR
 770  FA=FA/(RA-ROOT(JR))
```

```
      FB=FB/(RB-ROOT(JR))
      FR=FR/(RR-ROOT(JR))
C***       IF (ABS(FA).GT.1.D-50) GO TO 771
      IF (ABS(FA).GT.1.D-38) GO TO 771
C***       FA=FA*1.D50
      FA=FA*1.D38
      KA=KA+1
C***  771 IF(ABS(FB).GT.1.D-50) GO TO 772
  771 IF(ABS(FB).GT.1.D-38) GO TO 772
C***       FB=FB*1.D50
      FB=FB*1.D38
      KB=KB+1
C***  772 IF(ABS(FR).GT.1.D-50) GO TO 773
  772 IF(ABS(FR).GT.1.D-38) GO TO 773
C***       FR=FR*1.D50
      FR=FR*1.D38
      KR=KR+1
C      IF (ROOT(JR).LE.RC) NOV=NOV-1
  773 IF (ROOT(JR).LE.RC) NOV=NOV-1
      JR=JR+1
      NITE(JR)=0
      ROOT(JR)=RC
      IF (NOV.GT.0) GO TO 400
      NSK=0
C     ETA=2.0
C     GO TO 300
      GO TO 299
C
  900 NROOT=JR-1
      IF(NROOT.GT.0) GO TO 902
      WRITE (6,1180)
      STOP
  902 CONTINUE
      IF (IFPR.EQ.0) GO TO 905
      WRITE (6,1140)
      WRITE (6,1006) (NITE(J),J=1,NROOT)
      WRITE (6,1150)
      WRITE (6,1008) (TIM(J),J=1,NROOT)
      WRITE (6,1160)
      WRITE (6,1004) (ERRVL(J),J=1,NROOT)
      WRITE (6,1004) (ERRVR(J),J=1,NROOT)
C
C   READ EIGENVECTORS INTO CORE
  905 IF (NROOT.LE.NVM) GO TO 906
      NDIF=NROOT - NVM
      REWIND NT
      DO 904 L=1,NDIF
      READ (NT) (A(I,L),I=1,N)
  904 CONTINUE
      GO TO 908
  906 NDIF=0
  908 NROOT=NROOT - NDIF
      DO 912 L=1,NROOT
      DO 912 I=1,N
  912 A(I,L+NDIF)=VV(I,L)
```

```
C
C    ARRANGE EIGENVALUES AND VECTORS IN ASCENDING ORDER
        IF (JR.EQ.2) GO TO 950
        JR=JR-2
 910    IS=0
        DO 920 I=1,JR
        IF (ROOT(I+1).GE.ROOT(I)) GO TO 920
        IS=IS+1
        RT=ROOT(I+1)
        ROOT(I+1)=ROOT(I)
        ROOT(I)=RT
        DO 930 K=1,N
        RT=A(K,I+1)
        A(K,I+1)=A(K,I)
 930    A(K,I)=RT
 920    CONTINUE
        IF (IS.GT.0) GO TO 910
C
 950    WRITE (6,1170)
        NROOT=NSCH
        WRITE (6,1004) (ROOT(J),J=1,NROOT)
C
C     CALCULATE PHYSCIAL ERROR NORMS
C
      REWIND NT
      DO 955 L=1,NROOT
 955  WRITE (NT) (A(K,L),K=1,N)
      REWIND NSTIF
      READ (NSTIF) (A(I,1),I=1,NWA)
      REWIND NT
      DO 960 L=1,NROOT
      RT = ROOT(L)
      READ (NT) (V(I),I=1,N)
      CALL MULT (W,A,V,N,MA)
      VNORM=0.
      DO 958 I=1,N
 958  VNORM = VNORM + W(I)*W(I)
      DO 966 I=1,N
 966  W(I) = W(I) -RT*B(I)*V(I)
      WNORM = 0.0
      DO 968 I=1,N
 968  WNORM = WNORM + W(I)*W(I)
      VNORM =SQRT(VNORM)
      WNORM =SQRT(WNORM)
      ERRVL(L) = WNORM/VNORM
 960  CONTINUE
      REWIND NT
      DO 969 L=1,NROOT
 969  READ (NT) (A(K,L),K=1,N)
C
      WRITE (6,1190)
      WRITE ( 6,1004) (ERRVL(J),J=1,NROOT)
C
       REWIND NT
       DO 970 I=1,NROOT
```

```
  970 ROOT(I)=SQRT(ROOT(I))
       WRITE (NT) (ROOT(I),I=1,NROOT)
       NWA=N*NROOT
      WRITE (NT) (A(I,1),I=1,NWA)
      PI2=8.0D0*ATAN(1.0D0)
       DO 980 I=1,NROOT
 980   AT(I)=PI2/ROOT(I)
C
       RETURN
C
 1000  FORMAT (44H ***ERROR   NEG OR ZERO DIAGONAL ELEMENT A(,I4,4H) = ,
     1        E11.4,21HBEFORE DECOMPOSITION         )
 1004  FORMAT (1H0,6E20.12)
 1006  FORMAT (1H0,6I20)
 1008  FORMAT (1H0,6F20.2)
 1009  FORMAT (43H0***ERROR   SOLUTION TERMINATED IN *SECNTD*, /
     1        12X,25HRIGID BODY MODE(S) FOUND., / 1X)
 1010  FORMAT (51H1INVERSE ITERATION GIVES FOLLOWING APPROXIMATION TO,
     1        18H LOWEST EIGENVALUE, 1X)
 1015  FORMAT (41H0***ERROR   PRE-MATURE EXIT FROM *SECNTD*, / 12X,
     1        37HITERATION ABANDONED FOR ROOT NUMBER =, I4 / 12X,
     2        37HNUMBER OF ITERATIONS PERFORMED      =, I4 / 1X)
 1020  FORMAT (5H0RB = E20.12,7H NSCH = I4)
 1030  FORMAT (38H0***ERROR   SOLUTION STOP IN *SECANTD*, / 12X, 1H(,
     1        I3,48H) FACTORIZATIONS PERFORMED IN AN ATTEMPT TO FIND,
     2        32H LOWER BOUND ON FIRST EIGENVALUE, / 12X,
     3        16HCHECK THE MODEL., / 1X)
 1040  FORMAT (1H1,4X,4HROOT,4X,4HNITE,18X,2HRC,15X,12HDET (A-RC*B),15X,
     ./2HFC,13X,3HETA,4X,3HISC)
 1050  FORMAT (1H0,4X,I4,4X,I4,8X,3E22.14,F7.2,I6)
 1060  FORMAT (42H0THE DEFLATED POLYNOMIAL HAS NO MORE ROOTS  )
 1070  FORMAT (29H0(RC-RB) IS SMALLER THAN TOL )
 1080  FORMAT (16H0WE JUMPED OVER I4,16H UNKNOWN ROOT(S)  )
 1090  FORMAT (41H0***ERROR   PRE-MATURE EXIT FROM *SECNTD*,
     1        34H CAUSED BY EITHER OF THE FOLLOWING, / 12X,
     2        22H(1) BAD MODEL DATA, OR, / 12X,
     3        52H(2) ROOT CLUSTER (I.E., NEAR EQUAL OR REPEATED EIGEN,
     4        36HVALUES) ENCOUNTERED AT CURRENT SHIFT, / 16X,
     5        25HCAUSING STORAGE OVER-FLOW, 1X)
 1100  FORMAT (1H0,34X,4HROOT,18X,2HRQ,18X,4HNOR=,I2)
 1110  FORMAT (1H0,4X,I4,4X,I4,8X,2E22.14)
 1120  FORMAT (20H0TIME FOR INV ITERN F5.2)
 1140  FORMAT (42H0NO OF ITERATIONS FOR EACH EIGENVALUE ARE   /)
 1150  FORMAT (30H0TIME USED FOR EACH EIGENVALUE /)
 1160  FORMAT (43H0FOLLOWING ARE ERROR BOUNDS ON EIGENVALUES )
 1170  FORMAT (/// 40H WE SOLVED FOR THE FOLLOWING EIGENVALUES     )
 1180  FORMAT (37H0***ERROR   SOLUTION STOP IN *SECNTD*, / 12X,
     1        23HNO EIGENVALUES COMPUTED, / 1X)
 1190  FORMAT (/// 40H THE FOLLOWING ARE PHYSICAL ERROR BOUNDS,
     1             20H ON THE EIGENPAIRS     )
C
       END
      SUBROUTINE SELECT (MAT,NEL,T,TM,E,XNU,ALP,MAX,YM,PR,THERM)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
```

```
C
C
C     CALLED BY?  PIPEK
C
C     THIS ROUTINE SELECTS MATERIAL PROPERTIES FROM TABLES USING
C     LINEAR INTERPOLATION WITH TEMPERATURE
C
      DIMENSION TM(MAX,1),E(MAX,1),XNU(MAX,1),ALP(MAX,1)
C
C     IF THE TABLE HAS FEWER THAN MAX ENTRIES, THE TEMPERATURE VALUE
C     FOLLOWING THE LAST REAL ENTRY IS EQUAL TO -10000.0.  IF THE SECOND
C     TEMPERATURE POINT IS -10000.0, THE TABLE HAS ONE POINT, AND NO
C     INTERPOLATION IS PERFORMED.
C
      IF(MAX.LT.2) GO TO 5
      IF(TM(2,MAT).GT.-9999.) GO TO 10
    5 YM    =   E(1,MAT)
      PR    = XNU(1,MAT)
      THERM = ALP(1,MAT)
      RETURN
C
   10 DO 20 K=2,MAX
      IF(TM(K,MAT).LT.-9999.) GO TO 30
      N = K
      IF(T.GE.TM(K-1,MAT) .AND. T.LT.TM(K,MAT)) GO TO 40
   20 CONTINUE
C
   30 WRITE (6,3000) T,NEL,MAT
      STOP
C
   40 DT = TM(N,MAT) - TM(N-1,MAT)
      IF(DT.GT.1.0E-8) GO TO 50
      K = N-1
      WRITE (6,3010) K,N,MAT
      STOP
C
   50 RATIO = (T-TM(N-1,MAT))/ DT
      YM    =   E(N-1,MAT) + RATIO* (  E(N,MAT)-   E(N-1,MAT))
      PR    = XNU(N-1,MAT) + RATIO* (XNU(N,MAT)- XNU(N-1,MAT))
      THERM = ALP(N-1,MAT) + RATIO* (ALP(N,MAT)- ALP(N-1,MAT))
C
      RETURN
C
C
 3000 FORMAT (36HOERROR***  THE AVERAGE TEMPERATURE (,F12.3,5H) FOR,
     1 10H ELEMENT (,I4,1H), / 11X,28HCANNOT BE FOUND IN THE TABLE,
     2 22H FOR MATERIAL NUMBER (,I4,2H)., / 1X)
 3010 FORMAT (51HOERROR***  ZERO OR NEGATIVE TEMPERATURE DIFFERENCE ,
     1 16HBETWEEN POINTS (,I4, 7H) AND (,I4,1H), / 11X, 9HMATERIAL ,
     2 7HTABLE (,I4,2H)., / 1X)
C
      END
        SUBROUTINE SESOL (A,B,MAXA,NEQ,MA,NV,NBLOCK,NEQB,NAV,MI,NSTIF,
     1                   NRED,NL,NR)
C
```

```
C     CALLED BY?  SOLEQ
C
      DIMENSION A(NAV),B(NAV),MAXA(MI)
C
      MM=1
      MA2=MA - 2
      IF(MA2.EQ.0) MA2=1
      INC=NEQB - 1
      NWA=NEQB*MA
      NTB=(MA-2)/NEQB + 1
      NEB=NTB*NEQB
      NEBT=NEB + NEQB
      NWV=NEQB*NV
      NWVV=NEBT*NV
C
      N1=NL
      N2=NR
      REWIND NSTIF
      REWIND NRED
      REWIND N1
      REWIND N2
C
C     MAIN LOOP OVER ALL BLOCKS
      DO 600 NJ=1,NBLOCK
      IF (NJ.NE.1) GO TO 10
      READ (NSTIF) A
      IF(NEQ.GT.1) GO TO 100
      MAXA(1)=1
      WRITE(NRED) A,MAXA
      IF(A(1)) 1,174,3
    1 KK=1
      WRITE(6,1010) KK,A(1)
    3 DO 5 L=1,NV
    5 A(1+L)=A(1+L)/A(1)
      KR=1+NV
      WRITE(NL) (A(KK),KK=2,KR)
      RETURN
   10 IF (NTB.EQ.1) GO TO 100
      REWIND N1
      REWIND N2
      READ (N1) A
C
C     FIND COLUMN HEIGHTS
  100 KU=1
      KM=MINO(MA,NEQB)
      MAXA(1)=1
      DO 110 N=2,MI
      IF (N-MA) 120,120,130
  120 KU=KU + NEQB
      KK=KU
      MM=MINO(N,KM)
      GO TO 140
  130 KU=KU + 1
      KK=KU
      IF (N-NEQB) 140,140,136
```

```
 136    MM=MM - 1
 140    DO 160 K=1,MM
        IF (A(KK)) 110,160,110
 160    KK=KK - INC
 110    MAXA(N)=KK
C
        IF (A(1)) 172,174,176
 174    KK=(NJ-1)*NEQB + 1
        IF (KK.GT.NEQ) GO TO 590
        WRITE (6,1000) KK
        STOP
 172    KK=(NJ-1)*NEQB + 1
        WRITE (6,1010) KK,A(1)
C
C       FACTORIZE LEADING BLOCK
 176    DO 200 N=2,NEQB
        NH=MAXA(N)
        IF (NH-N) 200,200,210
 210    KL=N + INC
        K=N
        D=0.
        DO 220 KK=KL,NH,INC
        K=K - 1
        C=A(KK)/A(K)
        D=D + C*A(KK)
 220    A(KK)=C
        A(N)=A(N) - D
C
        IF (A(N)) 222,224,230
 224    KK=(NJ-1)*NEQB + N
        IF (KK.GT.NEQ) GO TO 590
        WRITE (6,1000) KK
        STOP
 222    KK=(NJ-1)*NEQB + N
        WRITE (6,1010) KK,A(N)
C
 230    IC=NEQB
        DO 240 J=1,MA2
        MJ=MAXA(N+J) - IC
        IF (MJ-N) 240,240,280
 280    KU=MIN0(MJ,NH)
        KN=N + IC
        C=0.
        DO 300 KK=KL,KU,INC
 300    C=C + A(KK)*A(KK+IC)
        A(KN)=A(KN) - C
 240    IC=IC + NEQB
C
        K=N + NWA
        DO 430 L=1,NV
        KJ=K
        C=0.
        DO 440 KK=KL,NH,INC
        KJ=KJ - 1
 440    C=C + A(KK)*A(KJ)
```

```
      A(K)=A(K) - C
 430  K=K + NEQB
C
 200  CONTINUE
C
C     CARRY OVER INTO TRAILING BLOCKS
      DO 400 NK=1,NTB
      IF ((NK+NJ).GT.NBLOCK) GO TO 400
      NI=N1
      IF ((NJ.EQ.1).OR.(NK.EQ.NTB)) NI=NSTIF
      READ (NI) B
      ML=NK*NEQB + 1
      MR=MINO((NK+1)*NEQB,MI)
     IF(MA.EQ.1) ML=MR
      MD=MI - ML
      KL=NEQB + (NK-1)*NEQB*NEQB
      N=1
C
      DO 500 M=ML,MR
      NH=MAXA(M)
      KL=KL + NEQB
      IF (NH-KL) 505,510,510
 510  K=NEQB
      D=0.
      DO 520 KK=KL,NH,INC
      C=A(KK)/A(K)
      D=D + C*A(KK)
      A(KK)=C
 520  K=K - 1
      B(N)=B(N) - D
      IF (MD) 580,580,530
 530  IC=NEQB
      DO 540 J=1,MD
      MJ=MAXA(M+J) - IC
      IF (MJ-KL) 540,550,550
 550  KU=MINO(MJ,NH)
      KN=N + IC
      C=0.
      DO 575 KK=KL,KU,INC
 575  C=C + A(KK)*A(KK+IC)
      B(KN)=B(KN) - C
 540  IC=IC + NEQB
C
 580  KN=N + NWA
      K=NEQB + NWA
      DO 610 L=1,NV
      KJ=K
      C=0.
      DO 620 KK=KL,NH,INC
      C=C + A(KK)*A(KJ)
 620  KJ=KJ - 1
      B(KN)=B(KN) - C
      KN=KN + NEQB
 610  K=K + NEQB
C
```

```
 505   MD=MD - 1
 500   N=N + 1
C
       IF (NTB.NE.1) GO TO 560
       WRITE (NRED) A,MAXA
       DO 570 I=1,NAV
 570   A(I)=B(I)
       GO TO 600
 560   WRITE (N2) B
C
 400   CONTINUE
C
       M=N1
       N1=N2
       N2=M
 590   WRITE (NRED) A,MAXA
C
 600   CONTINUE
C
C      VECTOR BACKSUBSTITUTION
       DO 700 K=1,NWVV
 700   B(K)=0.
       REWIND NL
C
       DO 800 NJ=1,NBLOCK
       BACKSPACE NRED
       READ (NRED) A,MAXA
       BACKSPACE NRED
       K=NEBT
       DO 810 L=1,NV
       DO 820 I=1,NEB
       B(K)=B(K-NEQB)
 820   K=K - 1
 810   K=K + NEBT + NEB
       KN=0
       KK=NWA
       NDIF=NEQB
       IF (NJ.EQ.1) NDIF=NEQB - (NBLOCK*NEQB - NEQ)
       DO 855 L=1,NV
       DO 850 K=1,NDIF
 850   B(KN+K)=A(KK+K)/A(K)
       KK=KK + NEQB
 855   KN=KN + NEBT
      IF(MA.EQ.1) GO TO 915
       ML=NEQB + 1
       KL=NEQB
       DO 860 M=ML,MI
       KL=KL + NEQB
       KU=MAXA(M)
       IF (KU-KL) 860,870,870
 870   K=NEQB
       KM=M
       DO 880 L=1,NV
       KJ=K
       DO 890 KK=KL,KU,INC
```

```
      B(KJ)=B(KJ)  - A(KK)*B(KM)
 890  KJ=KJ - 1
      KM=KM + NEBT
 880  K=K + NEBT
 860  CONTINUE
      N=NEQB
      DO 910 I=2,NEQB
      KL=N + INC
      KU=MAXA(N)
      IF (KU-KL) 910,920,920
 920  K=N
      DO 930 L=1,NV
      KJ=K
      DO 940 KK=KL,KU,INC
      KJ=KJ - 1
 940  B(KJ)=B(KJ)  - A(KK)*B(K)
 930  K=K + NEBT
 910  N=N - 1
C
  915 KK=0
      KN=0
      DO 950 L=1,NV
      DO 960 K=1,NEQB
      KK=KK + 1
 960  A(KK)=B(KN+K)
 950  KN=KN + NEBT
C
      WRITE (NL)  (A(K),K=1,NWV)
 800  CONTINUE
C
 1000 FORMAT (// 46H   STOP  ***  ZERO DIAGONAL ENCOUNTERED DURING,
     1              18H EQUATION SOLUTION, /
     2         13X,18H EQUATION NUMBER =, I6 )
 1010 FORMAT (/  50H WARNING ***  NEGATIVE DIAGONAL ENCOUNTERED DURING,
     1              18H EQUATION SOLUTION, /
     2         13X,18H EQUATION NUMBER =, I6, 5X, 7HVALUE =, E20.8 )
C
      RETURN
      END
      SUBROUTINE SHELL
C
C     CALLS?  TPLATE,STRSC
C     CALLED BY?  ELTYPE
C
      COMMON /one/ A(1)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /JUNK/ LT,LH,L,IPAD,SIG(20),IFILL(386)
      COMMON /EXTRA/ MODEX,NT8,N1OSV,NT10,IFILL2(12)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      IF (NPAR(1).EQ.0) GO TO 500
C     PROTECT NODAL TEMPERATURES
      N6=N5+NUMNP+12*NPAR(3)
      IF (N6.GT.MTOT) CALL ERROR (N6-MTOT)
```

```
C
      N6=N5+NUMNP
      CALL TPLATE(NPAR(2),NPAR(3),A(N1),A(N2),A(N3),A(N4),A(N6),NUMNP,
     1            MBAND)
C
      RETURN
C
  500 WRITE (6,2002)
      NUME=NPAR(2)
      numee=nume
      neqq=neq
      DO 800 MM=1,NUME
      CALL STRSC (A(N1),A(N3),NEQ,0)
      WRITE (6,2001)
      DO 800 L=LT,LH
      CALL STRSC (A(N1),A(N3),NEQ,1)
      WRITE (6,3002) MM,L,(SIG(I),I=1,6)
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,(SIG(I),I=1,6)
  800 CONTINUE
C
      RETURN
C
 2001 FORMAT (/)
 2002 FORMAT (24H1 SHELL ELEMENT STRESSES//
     1 '             ELEMENT      LOAD     MEMBRANE STRESS COMPONENTS
     2          BENDING MOMENT COMPONENTS', /
     3 '         NUMBER      CASE       SXX          SYY          S
     4XY          MXX          MYY          MXY',    //)
 3002 FORMAT (10X,2I10,6E12.4)
      END
C
C     CALLED BY?  QTSHEL
C
C     THIS SUBROUTINE FORMS THE PLATE BENDING STIFFNESS AND/OR THE
C     CONSISTENT LOAD VECTOR OF A LINEAR CURVATURE COMPATIBLE TRIANGLE
C     (LCCT) WITH 6, 5, 4 OR 3 NODAL POINTS
C
C
C * * * * * * * * * * * * * * * INPUTS  * * * * * * * * * * * * * * * *
C
C     M          NUMBER OF MIDPOINT DEGREES OF FREEDOM (M =3,2,1,0).
C                NOTE.. MIDPOINTS 4-5-6 (IF INCLUDED) ARE LOCATED ON
C                SIDES 2-3, 3-1 AND 1-2, RESPECTIVELY.
C
C     KKK        OPERATION FLAG
C                  KKK LE 0 = FORM STIFFNESS MATRIX AND LOAD VECTOR.
C                  KKK GT 0 = FORM LOAD VECTOR ONLY.
C
C  A(I),B(I)     I=1...3   PROJECTIONS OF SIDES 2-3, 3-1 AND 1-2 ONTO
C                X AND -Y, RESPECTIVELY.
C
C   C(I,J)       I=1...3, J=1...3   PLANE STRESS MATERIAL MATRIX.
C
```

```
C     H(I)        I=1...3   CORNER THICKNESSES (LINEAR VARIATION ASSUMED).
C
C     PT(I)       I=1...3   CORNER VALUES OF LATERAL DISTRIBUTED LOAD
C                 (LINEAR VARIATION ASSUMED).
C
C   BMT(I,J)      I=1...3, J=1...3   INITIAL BENDING MOMENT COMPONENTS
C                 MOM-XX (J=1), MOM-YY (J=2) AND MOM-XY (J=3) AT THE
C                 CORNERS I=1...3  (LINEAR VARIATION ASSUMED).
C
C
C * * * * * * * * * * * * * * * * OUTPUTS * * * * * * * * * * * * * * * * *
C
C   ST(I,J)       I=1..NDF, J=1..NDF  WITH NDF (NUMBER OF DOF) = 9+M, IS
C                 THE ELEMENT STIFFNESS MATRIX ASSOCIATED WITH THE NODAL
C                 DISPLACEMENT ORDERING
C                   W(1),RX(1),RY(1),W(2), .... RY(3),RM(1), ... RM(M)
C                 WHERE RM(1), ... RM(M), IF M GT 0, ARE MIDPOINT
C                 DEVIATIONS FROM NORMAL SLOPE LINEARITY
C
C    FT(I)        I=1..NDF   CONSISTENT NODAL FORCE VECTOR ASSOCIATED
C                 WITH THE NODAL DISPLACEMENT ORDERING DESCRIBED ABOVE.
C
C
      SUBROUTINE SLCCT (M,KKK)
      COMMON /TRIARG/ A(3),B(3), HMT(3), H(3),  C(3,3), SMT(3,3),
     1 BMT(3,3), FT(12),     PX(3),PY(3),PT(3),RM(3), ST(12,12)
      DIMENSION  P(21,12), G(21), Q(3,6), QB(3,6), T(3), U(3), HT(3),
     1 TX(3), TY(3), IPERM(3), XM(3,3), XMO(3)
      EQUIVALENCE (CM11,C(1)),(CM12,C(2)),(CM13,C(3)),(CM22,C(5)),
     1 (CM23,C(6)),(CM33,C(9))
      LOGICAL NOS, FLAT
      DATA IPERM/2,3,1/
      HO = (H(1)+H(2)+H(3))/3.
      IF (HO.LE.O.)  GO TO 1000
      NDF = 9 + M
      NOS = KKK.GT.O
      FLAT = (H(1).EQ.H(2)).AND.(H(2).EQ.H(3))
      AREA = A(3)*B(2)-A(2)*B(3)
      FAC = HO**3*AREA/864.
      PTF = AREA/6480.
      T(3) = 1.
      DO 150  I = 1,3
      J = IPERM(I)
      K = IPERM(J)
      X = A(I)**2+B(I)**2
      U(I)  = -(A(I)*A(J)+B(I)*B(J))/X
      X =SQRT(X)
      Y = 2.*AREA/X
      HT(I) =  2.*Y
      TX(I) =  Y*A(I)/X
      TY(I) = -Y*B(I)/X
      A1 = A(I)/AREA
      A2 = A(J)/AREA
      B1 = B(I)/AREA
      B2 = B(J)/AREA
```

```
      Q(1,I)   = B1*B1
      Q(2,I)   = A1*A1
      Q(3,I)   = 2.*A1*B1
      Q(1,I+3) = 2.*B1*B2
      Q(2,I+3) = 2.*A1*A2
      Q(3,I+3) = 2.*(A1*B2+A2*B1)
      DO 120  N = 1,3
  120 XM(N,I) = BMT(N,I)*AREA/72.
      IF (FLAT)  GO TO 150
      DO 140  N = 1,3
      L = IPERM(N)
      T(1) = H(N)/HO
      T(2) = H(L)/HO
      IF (T(1).GT.0.)  XM(N,I) = XM(N,I)/T(1)**3
      C1 = T(I)
      C2 = T(J)
      C3 = T(K)
      C4 = C2+C3
      C11 = C1*C1
      C23 = C2*C3
      C5 = C4*(3.*C1+C4) + 6.*C11 - 2.*C23
      C6 = C5 + 3.*C4*C4 - 4.*(C11+C23)
      QB(N,I  ) = (C1*(10.*C11-3.*C23)+C4*C5)/17.5 - 2.0
  140 QB(N,I+3) = (C1*(C11-2.*C23)+C4*C6)/35.0 - 1.0
  150 CONTINUE
      DO 200  I = 1,3
      J = IPERM(I)
      K = IPERM(J)
      II = 3*I
      JJ = 3*J
      KK = 3*K
      A1 = A(I)
      A2 = A(J)
      A3 = A(K)
      B1 = B(I)
      B2 = B(J)
      B3 = B(K)
      U1 = U(I)
      U2 = U(J)
      U3 = U(K)
      W1 = 1.-U1
      W2 = 1.-U2
      W3 = 1.-U3
      B1D = B1 + B1
      B2D = B2 + B2
      B3D = B3 + B3
      A1D = A1 + A1
      A2D = A2 + A2
      A3D = A3 + A3
      C21 = B1-B3*U3           + TX(K)
      C22 = -B1D+B2*W2+B3*U3 + TX(J)-TX(K)
      C31 = A1-A3*U3           + TY(K)
      C22 = -B1D+B2*W2+B3*U3 + TX(J)-TX(K)
      C31 = A1-A3*U3           + TY(K)
      C32 = -A1D+A2*W2+A3*U3 + TY(J)-TY(K)
```

```
      C51 = B3*W3-B2          + TX(K)
      C52 = B2D-B3*W3-B1*U1   + TX(I)-TX(K)
      C61 = A3*W3-A2          + TY(K)
      C62 = A2D-A3*W3-A1*U1   + TY(I)-TY(K)
      C81 = B3-B2D-B2*U2      + TX(J)
      C82 = B1D-B3+B1*W1      + TX(I)
      C91 = A3-A2D-A2*U2      + TY(J)
      C92 = A1D-A3+A1*W1      + TY(I)
      P1 = PT(I)*PTF
      P2 = PT(J)*PTF
      P3 = PT(K)*PTF
      U37 = 7.*U3
      W27 = 7.*W2
      W24 = 4.*W2
      U34 = 4.*U3
      C1 = 54.+W27
      C2 = 54.+U37
      C3 = 15.+W24
      C4 = 39.+U37
      C5 = 39.+W27
      C6 = 15.+U34
      TXS = TX(J)+TX(K)
      TYS = TY(J)+TY(K)
      FT(II-2) = 6.*((90.+U37+W27)*P1+(36.+U37+W24)*P2+(36.+U34+W27)*P3)
      FT(II-1) = (C1*B2-C2*B3+7.*TXS)*P1 + (C3*B2-C4*B3+4.*TXS+
     1 3.*TX(K))*P2 + (C5*B2-C6*B3+4.*TXS+3.*TX(J))*P3
      FT(II)   = (C1*A2-C2*A3+7.*TYS)*P1 + (C3*A2-C4*A3+4.*TYS+
     1 3.*TY(K))*P2 + (C5*A2-C6*A3+4.*TYS+3.*TY(J))*P3
      FT(K+9)  = (7.*(P1+P2)+4.*P3)*HT(K)
      XMO(I) = (XM(1,I)+XM(2,I)+XM(3,I))/3.
      DO 200  N = 1,3
      L = 6*(I-1) + N
      Q11 = Q(N,I)
      Q22 = Q(N,J)
      Q33 = Q(N,K)
      Q12 = Q(N,I+3)
      Q23 = Q(N,J+3)
      Q31 = Q(N,K+3)
      Q2333 = Q23-Q33
      Q3133 = Q31-Q33
      P(L   ,II-2) = 6.*(-Q11+W2*Q33+U3*Q2333)
      P(L   ,II-1) = C21*Q23+C22*Q33-B3D*Q12+B2D*Q31
      P(L   ,II  ) = C31*Q23+C32*Q33-A3D*Q12+A2D*Q31
      P(L   ,JJ-2) = 6.*(Q22+W3*Q2333)
      P(L   ,JJ-1) = C51*Q2333+B3D*Q22
      P(L   ,JJ  ) = C61*Q2333+A3D*Q22
      P(L   ,KK-2) = 6.*(1.+U2)*Q33
      P(L   ,KK-1) = C81*Q33
      P(L   ,KK  ) = C91*Q33
      P(L   ,I+9 ) = 0.
      P(L   ,J+9 ) = HT(J)*Q33
      P(L   ,K+9 ) = HT(K)*Q2333
      P(L+3 ,II-2) = 6.*(Q11+U3*Q3133)
      P(L+3 ,II-1) = C21*Q3133-B3D*Q11
      P(L+3 ,II  ) = C31*Q3133-A3D*Q11
```

```
      P(L+3 ,JJ-2) = 6.*(-Q22+U1*Q33+W3*Q3133)
      P(L+3 ,JJ-1) = C51*Q31+C52*Q33+B3D*Q12-B1D*Q23
      P(L+3 ,JJ  ) = C61*Q31+C62*Q33+A3D*Q12-A1D*Q23
      P(L+3 ,KK-2) = 6.*(1.+W1)*Q33
      P(L+3 ,KK-1) = C82*Q33
      P(L+3 ,KK  ) = C92*Q33
      P(L+3 ,I+9 ) = HT(I)*Q33
      P(L+3 ,J+9 ) = 0.
      P(L+3 ,K+9 ) = HT(K)*Q3133
      P(N+18,II-2) = 2.*(Q11+U3*Q12+W2*Q31)
      P(N+18,KK-1) = ((B1D-B2D)*Q33+C82*Q23+C81*Q31)/3.
      P(N+18,KK  ) = ((A1D-A2D)*Q33+C92*Q23+C91*Q31)/3.
  200 P(N+18,K+9 ) = HT(K)*Q12/3.
  300 DO 400  J = 1,NDF
      DO 340  L = 1,3
      II = L
      KK = L + 18
      P3 = P(KK,J)
      G(KK) = 0.
      DO 340  N = 1,3
      I = IPERM(N)
      JJ = II + 3
      P1 = P(II,J)
      P2 = P(JJ,J)
      SUM = P1 + P2 + P3
      G1 = SUM + P1
      G2 = SUM + P2
      G3 = SUM + P3
      IF (FLAT)  GO TO 320
      G1 = G1 + QB(N,1)*P1 + QB(N,6)*P2 + QB(N,5)*P3
      G2 = G2 + QB(N,6)*P1 + QB(N,2)*P2 + QB(N,4)*P3
      G3 = G3 + QB(N,5)*P1 + QB(N,4)*P2 + QB(N,3)*P3
  320 G(II) = G1
      G(JJ) = G2
      G(KK) = G3 + G(KK)
      II = II + 6
  340 FT(J) = FT(J) - XM(N,L)*G1 - XM(I,L)*G2 - XMO(L)*G3
      IF (NOS)  GO TO 400
      DO 360 N = 1,19,3
      G1 = G(N)
      G2 = G(N+1)
      G3 = G(N+2)
      G(N)   = CM11*G1 + CM12*G2 + CM13*G3
      G(N+1) = CM12*G1 + CM22*G2 + CM23*G3
  360 G(N+2) = CM13*G1 + CM23*G2 + CM33*G3
      DO 390  I = 1,J
      X = 0.
      DO 380  N = 1,21
  380 X = X + G(N)*P(N,I)
      X = X*FAC
      ST(I,J) = X
  390 ST(J,I) = X
  400 CONTINUE
 1000 RETURN
      END
```

```
C
C     CALLED BY?  QTSHEL
C
C     THIS SUBROUTINE FORMS THE PLANE STRESS STIFFNESS MATRIX AND/OR
C     THE CONSISTENT LOAD VECTOR OF A LINEAR STRAIN TRIANGLE (LST) WITH
C     6, 5 OR 4 NODAL POINTS, OR OF A CONSTANT STRAIN TRIANGLE (CST).
C     LINEAR ELASTIC ANISOTROPIC MATERIAL
C
C * * * * * * * * * * * * * * want INPUTS * * * * * * * * * * * * * * * *
C
C      M         NUMBER OF MIDPOINTS INCLUDED AS NODAL POINTS (M=3,2,1
C                FOR LST, M=0 FOR CST). NOTE.. MIDPOINTS 4-5-6 ARE
C                LOCATED ON THE SIDES 2-3, 3-1 AND 1-2, RESPECTIVELY.
C
C     KKK        OPERATION FLAG
C                  KKK LE 0 = FORM STIFFNESS MATRIX AND LOAD VECTOR.
C                  KKK GT 0 = FORM LOAD VECTOR ONLY.
C
C  A(I),B(I)     I=1...3   PROJECTIONS OF SIDES 2-3, 3-1 AND 1-2 ONTO
C                X AND -Y, RESPECTIVELY.
C
C   C(I,J)       I=1...3, J=1...3   PLANE STRESS MATERIAL MATRIX.
C
C    H(I)        I=1...3   CORNER THICKNESSES (LINEAR VARIATION ASSUMED).
C
C PX(I),PY(I)    I=1...3   CORNER VALUES OF X-Y COMPONENTS OF BODY FORCES
C                PER UNIT OF ELEMENT AREA (LINEAR VARIATION ASSUMED).
C
C   SMT(I,J)     I=1...3, J=1...3   INITIAL MEMBRANE STRESS COMPONENTS
C                SIG-XX (J=1), SIG-YY (J=2) AND SIG-XY (J=3 AT THE
C                CORNERS I=1,2,3  (LINEAR VARIATION ASSUMED).
C
C
C * * * * * * * * * * * * * * * OUTPUTS * * * * * * * * * * * * * * * *
C
C   ST(I,J)      I=1..NDF, J=1..NDF  WITH NDF (NUMBER OF DOF) = 6+2*M, IS
C                THE ELEMENT STIFFNESS MATRIX ASSOCIATED WITH THE NODAL
C                DISPLACEMENT ORDERING
C                    U(1),V(1),U(2),V(2),U(3), ..... V(3+M)
C                WHERE U(4), ... V(3+M), IF M GT 0, ARE DEVIATIONS
C                FROM LINEARITY AT THE MIDPOINTS 1...M.
C
C    FT(I)       I=1..NDF   CONSISTENT NODAL FORCE VECTOR ASSOCIATED
C                WITH THE NODAL DISPLACEMENT ORDERING DESCRIBED ABOVE.
C
C
      SUBROUTINE SLST (M,KKK)
      COMMON /TRIARG/ A(3),B(3), H(3),  HPT(3), C(3,3), SMT(3,3),
     1 BMT(3,3), FT(12),     PX(3),PY(3),PT(3),RM(3), ST(12,12)
      DIMENSION Q(3,3), QA(3), QB(3), A4(3), B4(3), IPERM(3),
     1 SXX(3),SYY(3), SXY(3)
      EQUIVALENCE (SXX(1),SMT(1)), (SYY(1),SMT(4)), (SXY(1),SMT(7))
C
      LOGICAL NOS
      DATA  IPERM /2,3,1/
```

```
      NOS = KKK.GT.0
      NDF = 6 + 2*M
      AREA = A(3)*B(2)-A(2)*B(3)
      SUMH = H(1)+H(2)+H(3)
      HO = SUMH/3.
      IF (HO) 500,500,140
  140 PXS = PX(1)+PX(2)+PX(3)
      PYS = PY(1)+PY(2)+PY(3)
      SXXH = 0.
      SYYH = 0.
      SXYH = 0.
      DO 150  I = 1,3
      CH = (SUMH+H(I))/24.
      SXXH = SXXH + CH*SXX(I)
      SYYH = SYYH + CH*SYY(I)
  150 SXYH = SXYH + CH*SXY(I)
      FAC = HO/(2.*AREA)
      C11 = C(1,1)*FAC
      C22 = C(2,2)*FAC
      C33 = C(3,3)*FAC
      C12 = C(1,2)*FAC
      C13 = C(1,3)*FAC
      C23 = C(2,3)*FAC
      DO 200  J = 1,3
      L = J + J
      FT(L-1)  =  (PXS+PX(J))*AREA/24. -  (B(J)*SXXH+A(J)*SXYH)
      FT(L)    =  (PYS+PY(J))*AREA/24. -  (A(J)*SYYH+B(J)*SXYH)
      IF (NOS)  GO TO 200
  180 DO 190  I = 1,J
      K = I + I
      AA = A(I)*A(J)
      BB = B(I)*B(J)
      AB = A(I)*B(J)
      BA = B(I)*A(J)
      ABA = AB+BA
      ST(K-1,L-1)  = C11*BB + C33*AA + C13*ABA
      ST(K  ,L  )  = C22*AA + C33*BB + C23*ABA
      ST(K-1,L  )  = C12*BA + C33*AB + C13*BB + C23*AA
  190 ST(K  ,L-1)  = C12*AB + C33*BA + C13*BB + C23*AA
  200 CONTINUE
      IF (M) 350,350,220
  220 DO 240  I = 1,3
      A4(I) = 4.*A(I)
      B4(I) = 4.*B(I)
      J = IPERM(I)
      K = IPERM(J)
      R = H(I)/HO
      Q(I,I) = 0.1+R/15.
      Q(J,K) = 0.1-R/60.
  240 Q(K,J) = Q(J,K)
      DO 300  J = 1,M
      J1 = IPERM(J)
      J2 = IPERM(J1)
      L = J + J + 6
      FX = 0.
```

```
      FY = 0.
      DO 250  N = 1,3
      Q1 = Q(N,J1)
      Q2 = Q(N,J2)
      QA(N)  = Q2*A4(J1)+Q1*A4(J2)
      QB(N)  = Q2*B4(J1)+Q1*B4(J2)
      FX = FX - QB(N)*SXX(N)-QA(N)*SXY(N)
  250 FY = FY - QA(N)*SYY(N)-QB(N)*SXY(N)
      FT(L-1)  =  (PXS-PX(J))*AREA/12. + FX*HO/2.
      FT(L)    =  (PYS-PY(J))*AREA/12. + FY*HO/2.
      IF  (NOS)  GO TO 300
      SUMQA = QA(1)+QA(2)+QA(3)
      SUMQB = QB(1)+QB(2)+QB(3)
      JM = J + 3
      DO 290  I = 1,JM
      K = I + I
      IF  (I.GT.3)  GO TO 260
      AA = A(I)*SUMQA
      AB = A(I)*SUMQB
      BA = B(I)*SUMQA
      BB = B(I)*SUMQB
      GO TO 280
  260 I1 = IPERM(I-3)
      I2 = IPERM(I1)
      AA = A4(I2)*QA(I1)+A4(I1)*QA(I2)
      AB = A4(I2)*QB(I1)+A4(I1)*QB(I2)
      BA = B4(I2)*QA(I1)+B4(I1)*QA(I2)
      BB = B4(I2)*QB(I1)+B4(I1)*QB(I2)
  280 ABA = AB+BA
      ST(K-1,L-1)  = C11*BB + C33*AA + C13*ABA
      ST(K  ,L  )  = C22*AA + C33*BB + C23*ABA
      ST(K-1,L  )  = C12*BA + C33*AB + C13*BB + C23*AA
  290 ST(K  ,L-1)  = C12*AB + C33*BA + C13*BB + C23*AA
  300 CONTINUE
  350 DO 400   I = 2,NDF
      DO 400   J = 1,I
  400 ST(I,J)  = ST(J,I)
  500 RETURN
      END
      SUBROUTINE SOLEIG
C
C     CALLS?  MODES,PRINTD
C     CALLED BY?  MAIN
C
C     SOLUTION OF THE EIGENVALUE PROBLEM
C
      COMMON /one/ A(1)
      COMMON /ELPAR/ NP(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /SOL/ NBLOCK,NEQB,LL,NF,IDUM,NEIG,NAD,NVV,ANORM,NFO
      COMMON /EM/ AT(1000),IFILL1(3138)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
      REAL TT(3)
      NT = 7
C
```

```
C     READ CONTROL CARD
C
C***        CALL TTIME (TT(1))
      WRITE (6,1003)
      READ (5,100) IFPR,IFSS,NITEM,RTOL,COFQ,NFO
C
      IF (IFPR.GT.0) IFPR=1
      IF (IFSS.GT.0) IFSS=1
      IF (NITEM.EQ.0) NITEM=16
      IF (RTOL.EQ.0.) RTOL=1.E-05
      IF (COFQ.EQ.0.) COFQ=1.E08
C
      IF (NEIG.GT.0) GO TO 10
C
      WRITE (6,1001)
      GO TO 15
   10 WRITE (6,1002)
   15 WRITE (6,1000) IFPR,IFSS,NITEM,RTOL,COFQ,NFO
 20   IF (MODEX.EQ.1) RETURN
      TPI=8.0D0*ATAN(1.0D0)
      COFQ=COFQ*TPI
      COFQ=COFQ*COFQ
C
C     CALL SOLUTION ROUTINE
C
      CALL MODES (NEQ,MBAND,NBLOCK,NEQB,NF,MTOT,IFPR,IFSS,RTOL,NITEM,
     1COFQ)
C
C***        CALL TTIME (TT(2))
C
C     WRITE CONTROL INFORMATION ON TAPE  -- FOR RESTART OPTION
C
      NC=2
      REWIND NC
      WRITE (NC) NEQ,NBLOCK,NEQB,MBAND,N1,NF,(AT(I),I=1,NF)
C
C     PRINT OF EIGENVALUES (OMEGA) AND EIGENVECTORS
C
      REWIND NT
      READ (NT) (A(I),I=1,NF)
      K=NF+1
      DO 30 I=1,NF
      K=K-1
      KK=(K-1)*3+1
      A(KK)=A(K)
      A(KK+1)=A(K)/TPI
 30   A(KK+2)=TPI/A(K)
      IF (NEIG.GT.0) GO TO 25
      WRITE (6,1009)
      DO 41 I=1,NF
      K1=3*I-2
      K2=3*I
 41   WRITE (6,1020) I,(A(J),J=K1,K2)
      GO TO 35
   25 WRITE (6,1010)
```

```
      DO 40 I=1,NF
      K1=3*I-2
      K2=3*I
   40 WRITE (6,1020) I,(A(J),J=K1,K2),AT(NF+1)
C
   35 N1=1
      N2=N1+NUMNP*6
      N3=N2+6*NF
      WRITE (6,1030)
      CALL PRINTD (A(N1),A(N2),A(N3),NEQB,NUMNP,NF,NBLOCK,NEQ,NT,2)
C***      CALL TTIME (TT(3))
C
C     COMPUTE TIME LOG
C
      DO 50 K=1,2
   50 TT(K) = TT(K+1)-TT(K)
      WRITE (6,2000) (TT(L),L=1,2)
C
  100 FORMAT (3I5,2F10.0,I5)
C
 1000 FORMAT (1H0 // 20H CONTROL INFORMATION, //
     1         5X,31HFLAG FOR ADDITIONAL PRINTING  =, I5 /
     2         7X,14HEQ.0, SUPPRESS, /
     3         7X,11HEQ.1, PRINT,    //
     4         5X,31HSTURM SEQUENCE CHECK FLAG (*) =, I5 /
     5         7X,19HEQ.0, PERFORM CHECK, /
     6         7X,10HEQ.1, PASS,          //
     7         5X,31HMAXIMUM ITERATION CYCLES (*)  =, I5 //
     8         5X,31HCONVERGENCE TOLERANCE (*)     =, E14.4 //
     9         5X,31HCUT-OFF FREQUENCY (CPS)       =, E14.4 //
     .         5X,31HNUMBER OF STARTING ITERATION   ,       /
     .         5X,31HVECTORS TO BE READ FROM        ,       /
     .         5X,31HTAPE10 (*)                    =, I5    ///
     A         5X,27H(*)  APPLICABLE TO SUBSPACE, /
     B         5X,29H     ITERATION SOLUTIONS ONLY, 1X)
 1001 FORMAT (44H0DETERMINANT SEARCH SOLUTION IS CARRIED OUT  )
 1002 FORMAT (44H0SUBSPACE ITERATION SOLUTION IS CARRIED OUT  )
 1003 FORMAT (1H1,//41H E I G E N V A L U E   A N A L Y S I S   //)
 1009 FORMAT (1H1,'20HPRINT OF FREQUENCIES',//
     1              '23H MODE      CIRCULAR  ', /
     2              '49H NUMBER    FREQUENCY    FREQUENCY    PERIOD',/
     3             ' 49H           (RAD/SEC)   (CYCLES/SEC)    (SEC)')
 1010 FORMAT (1H1,'PRINT OF FREQUENCIES',//
     1           '  23H MODE       CIRCULAR    ',/
     2            ' 58H NUMBER     FREQUENCY    FREQUENCY    PERIOD    TOL
     3ERANCE  ',  /
     4           '   49H           (RAD/SEC)   (CYCLES/SEC)    (SEC) '  )
 1020 FORMAT (1H0,I4,6X,4(E10.4,2X))
 1030 FORMAT (/// 22H PRINT OF EIGENVECTORS, // 1X)
 2000 FORMAT (//// 44H E I G E N S O L U T I O N   T I M E   L O G,
     1        //5X,15HEIGENSOLUTION =, F8.2 /
     2          5X,15HPRINTING      =, F8.2 /)
C
      RETURN
      END
```

```
      SUBROUTINE SOLSTP (IDIS,ISTR,MASS,B,XO,X1,X2,A,MAXA,SDIS,SSTR,
     1                   NSD,NSS,NEQ,NEQB,MBAND,NWA,MI,MM,NBLOCK)
      REAL MASS
C
C     CALLS?  REDVK
C     CALLED BY?  STEP
C
C     THIS ROUTINE SOLVES FOR DISPLACEMENTS, VELOCITIES AND ACCELERA-
C     TIONS AT EACH SOLUTION TIME STEP AND SAVES ONLY THOSE DISPLACEMENT
C     COMPONENTS REQUIRED FOR EITHER HISTORY PRINTING OR STRESS HISTORY
C     RECOVERY.
C
C     *TAPE2*  CONTAINS LOAD VECTORS FOR EACH TIME STEP
C     *TAPE3*  CONTAINS THE REDUCED EFFECTIVE STIFFNESS MATRIX
C              IN BLOCK FORM
C     *TAPE4*  USED TO SAVE DISPLACEMENTS FOR HISTORY OUTPUT     *JT*
C     *TAPE7*  USED TO SAVE DISPLACEMENTS FOR STRESS RECOVERY    *IT*
C
      DIMENSION      IDIS(NSD),ISTR(NSS),MASS(NEQ),B(NEQ),XO(NEQ),
     1               X1(NEQ),X2(NEQ),A(NWA),MAXA(MI),SDIS(MM,NSD),
     2               SSTR(MM,NSS),ISAVE(3)
C
      COMMON /DYN/   NT,NOT,ALFA,DELT,BETA,IFILL1(4)
      COMMON /EXTRA/ MODEX,NT8,N1OSV,NT10,IFILL2(12)
C
C     SET TAPE ASSIGNMENTS
C
      JT = 4
      IT = 7
      KT = 2
C
      REWIND JT
      REWIND KT
      REWIND IT
      REWIND NT10
C
C     SET FLAGS FOR SAVING SYSTEM (DIS/VEL/ACC) VECTORS
C
      I = N1OSV
      L = 4
      DO 50 K=1,3
      L = L-1
      ISAVE(L) = I - I/10*10
   50 I = I/10
C
C     CLEAR SYSTEM VECTORS (I.E., ZERO INITIAL CONDITIONS ASSUMED)
C
      DO 100 I=1,NEQ
      XO(I)=0.0
      X1(I)=0.0
  100 X2(I)=0.0
C
C     COMPUTE THE TIME CONSTANTS FOR INTEGRATION
C
      TETA=1.4
```

```
      DELT1=TETA*DELT
      DELT2=DELT1**2
      AO=(6.+3.*ALFA*DELT1)/(DELT2+3.*BETA*DELT1)
      BO=ALFA-BETA*AO
      A1=6./DELT2+3.*BO/DELT1
      A2=6./DELT1+BO+BO
      A3=2.+BO*DELT1/2.
      A4=6./(3.*BETA*DELT1+DELT2)/TETA
      B1=BETA*A4
      A5=3.*B1/DELT1-6./DELT2/TETA
      A6=2.*B1-6./DELT1/TETA
      A7=.5*B1*DELT1+1.-3./TETA
      A8=0.5*DELT
      A9=DELT**2/3.0
      A10=0.5*A9
C
C     T I M E   S T E P   L O O P
C
      IK=0
C
      DO 700 K=1,NT
C
C     READ THE VECTOR OF APPLIED FORCES AT THIS SOLUTION STEP
C
      READ (KT) B
C
C     COMPUTE THE EFFECTIVE LOAD VECTOR
C
      DO 450 I=1,NEQ
  450 B(I)=B(I)+MASS(I)*(A1*XO(I)+A2*X1(I)+A3*X2(I))
C
C     SOLVE FOR DISPLACEMENT VECTOR
C
      CALL REDVK (A,B,MAXA,NEQB,NWA,NEQ,NBLOCK,MI,MBAND,K)
C
C     COMPUTE DISPLACEMENTS  *XO*    *
C             VELOCITIES     *X1*    *  AT TIME STEP *K*
C             ACCELERATIONS  *X2*    *
C
      DO 500 I=1,NEQ
      ACC=A4*B(I)+A5*XO(I)+A6*X1(I)+A7*X2(I)
      XO(I)=XO(I)+DELT*X1(I)+A9*X2(I)+A10*ACC
      X1(I)=X1(I)+A8*(X2(I)+ACC)
  500 X2(I)=ACC
C
C     PERFORM TAPE SAVE OPERATIONS ON SYSTEM VECTORS AT TIME STEP, K.
C
      IF(N10SV.LT.1) GO TO 520
C
      IF(ISAVE(1).LT.1) GO TO 510
      I = K -K/ISAVE(1)*ISAVE(1)
      IF(I.EQ.0)
     *WRITE (NT10) XO
  510 IF(ISAVE(2).LT.1) GO TO 515
      I = K -K/ISAVE(2)*ISAVE(2)
```

```
      IF(I.EQ.0)
     *WRITE (NT10) X1
  515 IF(ISAVE(3).LT.1) GO TO 520
      I = K -K/ISAVE(3)*ISAVE(3)
      IF(I.EQ.0)
     *WRITE (NT10) X2
C
  520 CONTINUE
C
C     TEST TO SEE IF DISPLACEMENTS ARE TO BE SAVED FOR PRINTING OR
C     ELEMENT STRESS RECOVERY
C
      L = K - K/NOT*NOT
      IF(L.NE.0) GO TO 700
      IK=IK+1
      IF(NSD.LT.1) GO TO 610
      DO 600 I=1,NSD
      J=IDIS(I)
  600 SDIS(IK,I)=X0(J)
  610 IF(NSS.LT.1) GO TO 660
      DO 650 I=1,NSS
      J=ISTR(I)
  650 SSTR(IK,I)=X0(J)
  660 IF(IK.NE.MM) GO TO 700
      IK=0
      IF(NSD.GT.0) WRITE (JT) SDIS
      IF(NSS.GT.0) WRITE (IT) SSTR
C
  700 CONTINUE
C
C     E N D   O F   T I M E   M A R C H I N G   L O O P
C
      IF(IK.EQ.0) RETURN
      IF(NSD.GT.0) WRITE (JT) SDIS
      IF(NSS.GT.0) WRITE (IT) SSTR
C
      RETURN
      END
      SUBROUTINE SOL21
C
C     CALLED BY ? ELTYPE
C     CALLS ? STRSC
C
C
C     3 / D 8 TO 21 NODE SOLID ELEMENTS
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/ NS,ND,LM(63)
      COMMON/JUNK/ LT,LH,L,N6,SIG(42),N7,N8,N9,N10,N11,N12,N13,N14,
     1         N15,N16,N17
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      COMMON /one/  A(1)
```

```
C
C
      IF(NPAR(1).EQ.0) GO TO 500
C
C     ERROR CHECKS AND SET DEFAULT VALUES IF REQUIRED
C
      WRITE (6,1000)
      IF(NPAR(2).GT.0) GO TO 10
      WRITE (6,1001) (NPAR(K),K=1,10)
      WRITE (6,1002)
      STOP
   10 IF(NPAR(3).GT.0) GO TO 20
      WRITE (6,1001) (NPAR(K),K=1,10)
      WRITE (6,1003)
      STOP
   20 IF(NPAR(4).EQ.0) NPAR(4) = 1
      IF(NPAR(7).EQ.0) NPAR(7) = 21
      IF(NPAR(7).GE.8 .AND. NPAR(7).LE.21) GO TO 30
      WRITE (6,1001) (NPAR(K),K=1,10)
      WRITE (6,1004)
      STOP
   30 IF(NPAR(9).EQ.0) NPAR(9) = 2
      IF(NPAR(9).GE.2 .AND. NPAR(9).LE.4) GO TO 40
      WRITE (6,1001) (NPAR(K),K=1,10)
      WRITE (6,1005)
      STOP
   40 IF(NPAR(10).EQ.0) NPAR(10) = 2
      IF(NPAR(10).GE.2 .AND. NPAR(10).LE.4) GO TO 50
      WRITE (6,1001) (NPAR(K),K=1,10)
      WRITE (6,1005)
      STOP
C
C     STORAGE ALLOCATION
C
C     A(N6)  = STARTING LOCATION OF WEIGHT DENSITY
C     A(N7)  = STARTING LOCATION OF MASS DENSITY
C     A(N8)  = STARTING LOCATION OF VECTOR CONTAINING THE ACTUAL
C               NUMBER OF TEMPERATURE POINTS FOR EACH MATERIAL TABLE
C     A(N9)  = STARTING LOCATION OF MATERIAL PROPERTY TABLE
C     A(N10) = STARTING LOCATION OF DIRECTION COSINE ARRAYS FOR
C               MATERIAL ORIENTATION AXIS
C     A(N11) = STARTING LOCATION OF SURFACE LOAD FACE NUMBERS
C     A(N12) = STARTING LOCATION OF SURFACE LOAD CODE TYPES
C     A(N13) = STARTING LOCATION OF PRESSURE WORKING ARRAY
C     A(N14) = STARTING LOCATION OF OUTPUT REQUEST LOCATION SETS
C     A(N15) = STARTING LOCATION OF VECTOR CONTAINING THE ACTUAL
C               NUMBER OF REQUESTED OUTPUT LOCATION IN EACH OUTPUT SET
C     A(N16) = STARTING LOCATION OF ELEMENT STIFFNESS MATRIX
C
   50 N6 = N5 + NUMNP
      N7 = N6 + NPAR(3)
      N8 = N7 + NPAR(3)
      N9 = N8 + NPAR(3)
      N10 = N9 + NPAR(3) * NPAR(4) * 13
      N11 = N10 + NPAR(5) * 9
```

```
      N12 = N11 + NPAR(6)
      N13 = N12 + NPAR(6)
      N14 = N13 + NPAR(6) * 7
      N15 = N14 + NPAR(8) * 7
      N16 = N15 + NPAR(8)
      N17 = N16 + NPAR(7) * 189
C
      IF(N17.GT.MTOT) CALL ERROR(N17-MTOT)
C
C     PROCESS ELEMENT DATA, AND GENERATE ELEMENT MATRICES
C
      CALL THDFE  (A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),A(N9),
     1            A(N10),A(N11),A(N12),A(N13),A(N14),A(N15),A(N16),
     2            NPAR(2),NPAR(3),NPAR(4),NPAR(5),NPAR(6),NPAR(7),
     3            NPAR(8),NPAR(9),NPAR(10),NUMNP)
C
      RETURN
C
C     RECOVER ELEMENT STRESSES (STATIC CASES ONLY)
C
  500 WRITE (6,2001)
      NUME = NPAR(2)
C
      DO 800 MM=1,NUME
C
C
C***  STRESS PORTHOLE
      CALL STRSC (A(N1),A(N3),NEQ,0)
      IF(N10SV.EQ.1)
     *WRITE (NT10) NS
C***
C
      IF(NS.EQ.1) GO TO 800
C
      WRITE (6,5000)
C
      DO 700 L=LT,LH
C
C
      CALL STRSC (A(N1),A(N3),NEQ,1)
      LOC = NS/6
      K1 = -5
C
      DO 600 N=1,LOC
      K1 = K1 + 6
      K2 = K1 + 5
C
      IF(N.EQ.1) WRITE (6,3001) MM,L,N,(SIG(I),I=K1,K2)
      IF(N.GT.1) WRITE (6,4001)  N,(SIG(I),I=K1,K2)
C
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,N,(SIG(I),I=K1,K2)
C***
  600 CONTINUE
```

```
C
      WRITE (6,5000)
C
  700 CONTINUE
  800 CONTINUE
      RETURN
C
C     FORMATS
C
 1000 FORMAT (53H121 - N O D E  S O L I D  E L E M E N T  I N P U T   ,
     1 10HD A T A   ,//38HOCONTROL INFORMATION                     ,/1X)
 1001 FORMAT (48HOERROR DETECTED WHILE PROCESSING MASTER ELEMENT ,
     1 12HCONTROL CARD,//16X,1H(,10I5,1H),/1X)
 1002 FORMAT (32H NO 3/D SOLID ELEMENTS SPECIFIED,/1X)
 1003 FORMAT (23H NO MATERIALS REQUESTED, / 1X)
 1004 FORMAT (49H MAXIMUM NUMBER OF NODES MUST BE GE.8 .AND. LE.21,/1X)
 1005 FORMAT (42H INTEGRATION ORDER MUST BE GE.2 .AND. LE.4,/1X)
 2001 FORMAT (54H121 - N O D E  S O L I D  E L E M E N T  S T R E S S
     .  //
     .23H ELEMENT  LOAD LOCATION,9X,6HSIG-XX,9X,6HSIG-YY,9X,6HSIG-ZZ,
     3 9X,6HSIG-XY,9X,6HSIG-YZ,9X,6HSIG-ZX,//1X)
 3001 FORMAT (I8,I6,I9,6E15.6)
 4001 FORMAT (  14X,I9,6E15.6)
 5000 FORMAT ( / )
C
      END
      SUBROUTINE SPECTR  (F,PX,XM,W,MASS,NEQB,NF,NBLOCK,TM)
C
C     CALLS?  SD
C     CALLED BY?  RESPEC
C
      COMMON /EXTRA/ MODEX,NT8,IFILL(14)
      DIMENSION PX(NF,3),F(NEQB,NF),XM(NEQB),W(NF),MASS(NEQB)
      DIMENSION DIRN(3)
C
C     COMPUTES MODAL AND R.M.S. DISPL RESPONSE TO EARTHQUAKE
C
      IF (MODEX.EQ.1) GO TO 270
      TPI=6.2831853
      DO 100 I=1,NF
      DO 100 J=1,3
  100 PX(I,J)=0.
C
C      FORM MODAL PARTICIPATION FACTORS PX(I,IDRN)
C           IDRN=1,2,3 .... FOR X,Y,Z, DIRN EARTHQUAKE
C
      REWIND 9
      REWIND 3
      DO 200 N=1,NBLOCK
      BACKSPACE 7
      READ (7) F
      BACKSPACE 7
      READ (3) MASS
      READ (9) XM
C
```

```
      DO 250 I=1,NEQB
      J=MASS(I)
      IF (J.LE.0) GO TO 250
      DO 240 L=1,NF
  240 PX(L,J)=PX(L,J)+F(I,L)*XM(I)
  250 CONTINUE
  200 CONTINUE
C
C     READ FREQUENCIES W OFF TAPE 7
C
      BACKSPACE 7
      READ (7) W
      REWIND 2
      WRITE (2) W
C
C     COMPUTE MODAL AMPLITUDES (IN W) FROM SPECTRUM AND PX
C
  270 READ (5,1000) DIRN,IND
      WRITE (6,2000) DIRN
      WRITE (6,2010) IND
      IF (MODEX.EQ.1) W(1)=SD(1)
      IF (MODEX.EQ.1) RETURN
      WRITE (6,2020)
      DO 280 I=1,NF
  280 WRITE (6,2040) I,(PX(I,J),J=1,3)
      DO 300 I=1,NF
      WW=TPI/W(I)
      WR=0.
      DO 290 K=1,3
  290 WR=WR +ABS(PX(I,K))*DIRN(K)
      WR=WR*SD(WW)
      IF (IND.EQ.1) WR=WR/(W(I)*W(I))
  300 W(I)=WR
C
C     WRITE MODAL DISPLS F AND R.M.S. ON TAPE 2
C
      REWIND 7
      READ (7)
      DO 350 N=1,NBLOCK
      READ (7) F
      DO 310 J=1,NF
      AMP=W(J)
      DO 310 I=1,NEQB
  310 F(I,J)=F(I,J)*AMP
C
      DO 320 I=1,NEQB
      WW=0.
      DO 330 J=1,NF
  330 WW=WW+F(I,J)**2
  320 XM(I)=SQRT(WW)
  350 WRITE (2) F,XM
C
      RETURN
 1000 FORMAT (3F10.0,I5)
 2000 FORMAT (20H DIRECTION FACTORS   / /
```

```
     1          10X,3HX = F10.4,4X,3HY = F10.4,4X,3HZ = F10.4   //)
 2010 FORMAT (56HOINDICATOR FOR DISPLACEMENT OR ACCELERATION SPECTRUM =
     1 I5  //
     2          20H EQ.0  DISPLACEMENT     /
     3          20H EQ.1  ACCELERATION     ///)
 2020 FORMAT (28H MODAL PARTICIPATION FACTORS, // 5H MODE,3X,
     1 11HX-DIRECTION,3X,11HY-DIRECTION,3X,11HZ-DIRECTION, / 1X)
 2040 FORMAT (1H ,I4,3E14.4 / 1X)
      END
      SUBROUTINE SPLOT (IT,JT,NDS,ISP)
C
C     CALLED BY?  SDSPLY
C
C     ROUTINE TO PRODUCE PRINTER PLOTS OF TIME HISTORIES,
C     EIGHT (MAXIMUM) TRACES PER PLOT.
C
C
      COMMON /EM/     PP(101),KD(2,8),XM(8),TM(8),IP(8),X(8),IFILL1(4864)
      COMMON /DYN/    NT,NOT,DAMP,DT,BETA,IFILL2(4)
C
      DIMENSION       SM(8)
C
      DATA            SM/1H1,1H2,1H3,1H4,1H5,1H6,1H7,1H8/
      DATA            BL/1H /,V/1HX/,AST/1H*/
C
      READ (IT) KD,XM,TM,L
      WRITE (6,3000) (KD(1,I),KD(2,I),XM(I),TM(I),I,I=1,L)
C
      DO 100 I=1,L
      IF(XM(I)) 50,100,50
   50 XM(I)=50./XM(I)
  100 CONTINUE
      TT=0.
      WRITE (6,999)
      WRITE (6,1000)
      WRITE (6,2000) TT,(V,I=1,101),TT
C
      K=1
      DO 200 I=2,100
  200 PP(I)=BL
C
C     CONSIDER EACH OUTPUT STEP
C
      DO 500 N=1,NDS
      READ (JT) X
      PP(1)=V
      PP(51)=V
      PP(101)=V
C
  220 II=ISP
  210 IF(II.LE.0) GO TO 250
      WRITE (6,2001) PP
      II=II-1
      GO TO 210
C
```

```
  250 TT=TT+DT
      DO 300 I=1,L
      XX=XM(I)*X(I)
      M=XX
      M=M+51
      IP(I)=M
      IF(PP(M).EQ.V .OR. PP(M).EQ.BL) GO TO 270
      PP(M) = AST
      GO TO 300
  270 PP(M) = SM(I)
  300 CONTINUE
      IF(K.LT.10)  GO TO 320
      K=1
      WRITE (6,2000) TT,PP,TT
      GO TO 340
  320 WRITE (6,2001) PP
      K=K+1
C
C     RESET PP
C
  340 DO 360 I=1,L
      M=IP(I)
  360 PP(M)=BL
  500 CONTINUE
      TT=TT+DT
C
      WRITE (6,2000) TT,(V,I=1,101),TT
      WRITE (6,1000)
C
      RETURN
C
C     F O R M A T S
C
  999 FORMAT (1H1,57X,15HO R D I N A T E )
 1000 FORMAT ( / 1H ,3X,7HT I M E,2X,4H-1.0,21X,4H-0.5,22X,3H0.0,22X,
     1          3H0.5,22X,3H1.0,4X,7HT I M E, 1X)
 2000 FORMAT (1H ,F10.4,4X,101A1,F12.4)
 2001 FORMAT (1H ,14X,101A1)
 3000 FORMAT (I8,12X,I3,1P2E14.4,3X,I6)
C
      END
      SUBROUTINE SSLAW (D,E,TEMP,DCA,KAXES,KMAT,NEL,DUM,ALPHA)
C
C     CALLED BY ? THDFE
C
C
C     THIS ROUTINE FORMS THE STRESS-STRAIN LAW IN MATERIAL COORDINATES
C     (X1,X2,X3) AND TRANSFORMS THE MATERIAL SYSTEM LAW TO GLOBAL
C     COORDINATES (X,Y,Z).
C
      DIMENSION D(6,6),E(12),TEMP(6,6),DCA(3,3),IPRM(3),DUM(6,6),
     1         ALPHA(6)
C
      DATA IPRM / 2,3,1 /
C
```

```
C     FORM THE DIRECT STRAIN PARTITION OF THE STRAIN-STRESS LAW IN
C     MATERIAL COORDINATES (X1,X2,X3)
C
      DO 20 I=1,3
      ALPHA(I)   = E(I+9)
      ALPHA(I+3) = 0.0
      IF(E(I).GT.1.0E-08) GO TO 15
      WRITE (6,3000) I,I,KMAT,NEL
      STOP
 3000 FORMAT (23HOERROR***   MODULUS E(,2I1,16H) FOR MATERIAL (,I3,
     1          14H) IN ELEMENT (,I5,10H) IS ZERO., / 1X)
   15 TEMP(I,I) = 1.0/E(I)
   20 CONTINUE
C
      TEMP(1,2) = -E(4)* TEMP(2,2)
      TEMP(2,1) =        TEMP(1,2)
      TEMP(1,3) = -E(5)* TEMP(3,3)
      TEMP(3,1) =        TEMP(1,3)
      TEMP(2,3) = -E(6)* TEMP(3,3)
      TEMP(3,2) =        TEMP(2,3)
C
C     INVERT THE DIRECT STRAIN PARTITION
C
      DO 60 N=1,3
      X = 1.0/TEMP(N,N)
      DO 30 J=1,3
   30 TEMP(N,J) = - TEMP(N,J)* X
C
      DO 50 I=1,3
      IF(N.EQ.I) GO TO 50
      DO 40 J=1,3
      IF(N.EQ.J) GO TO 40
      TEMP(I,J) = TEMP(I,J) + TEMP(I,N) * TEMP(N,J)
   40 CONTINUE
   50 TEMP(I,N) = TEMP(I,N) * X
C
      TEMP(N,N) = X
   60 CONTINUE
C
C     FORM THE COMPLETE STRESS-STRAIN LAW IN MATERIAL COORDINATES
C
      DO 70 I=1,6
      DO 70 J=1,6
   70 D(I,J) = 0.0
C
      DO 80 I=1,3
      DO 80 J=1,3
   80 D(I,J) = TEMP(I,J)
C
      D(4,4) =E(7)
      D(5,5) = E(9)
      D(6,6) = E(8)
C
C     TRANSFORM THE MATERIAL LAW TO GLOBAL COORDINATES (X,Y,Z)
C
```

```
      IF(KAXES.LT.1) RETURN
C
C     TRANSFORMATION BETWEEN MATERIAL STRAINS AND GLOBAL STRAINS
C
      DO 100 I1=1,3
      I2 = IPRM(I1)
      DO 90 J1 = 1,3
      J2 = IPRM(J1)
      TEMP(I1  ,J1  ) = DCA(J1,I1)*DCA(J1,I1)
      TEMP(I1+3,J1  ) = DCA(J1,I1)*DCA(J1,I2) * 2.0
      TEMP(I1  ,J1+3) = DCA(J1,I1)*DCA(J2,I1)
      TEMP(I1+3,J1+3) = DCA(J1,I1)*DCA(J2,I2) +
     1                  DCA(J2,I1)*DCA(J1,I2)
   90 CONTINUE
  100 CONTINUE
C
C     ROTATE THE MATERIAL LAW TO THE GLOBAL SYSTEM
C
      DO 130 I=1,6
      DO 120 J=1,6
      X = 0.0
      DO 110 K=1,6
  110 X = X + D(I,K)*TEMP(K,J)
  120 DUM(I,J) = X
  130 CONTINUE
C
      DO 160 I=1,6
      DO 150 J=I,6
      X = 0.0
      DO 140 K=1,6
  140 X = X + TEMP(K,I)*DUM(K,J)
      D(I,J) = X
      D(J,I) = X
  150 CONTINUE
  160 CONTINUE
C
C     TRANSFORM THE EXPANSION COEFFICIENTS FROM MATERIAL COORDINATES
C     TO GLOBAL COORDINATES
C
C
      DO 200 I=1,6
      X = 0.0
      DO 190 K=1,3
  190 X = X + TEMP(K,I)*E(K+9)
      IF(I.GT.3) X =X*2.0
  200 ALPHA(I) = X
C
      RETURN
      END
       SUBROUTINE  SSPCEB (NEQ,MBAND,NBLOCK,NEQB,NF,NV,NWA,NWV,NWVV,NTB,
     1IFPR,IFSS,NITEM,RTOL,ANORM,COFQ)
      REAL TIM1,TIM2,TIM3
C
C     CALLS?  INVECT,DECOMP,REDBAK,EIGSOL,SCHECK
C     CALLED BY?  MODES
```

```
C
      COMMON /TAPES/NSTIF,NRED,NL,NR,NT,NMASS
      COMMON /EM/ AT(1000),IFILL(3138)
      COMMON /one/ A(1)
C
C     ESTABLISH STARTING TRANSFORMATION VECTORS ON TAPE NR
C
      N2=1+NWV
      N3=N2+NEQB
C***      CALL TTIME(TIM1)
      CALL INVECT(A(1),A(N2),A(N3),NBLOCK,NEQB,NV,IFPR)
C***      CALL TTIME(TIM2)
C
C   FACTORIZE STIFFNESS MATRIX
      N2=1+NWA
      N3=N2+NWA
      MI = MBAND + NEQB - 1
      CALL DECOMP  (A(1),A(N2),A(N3),NEQB,MBAND,NBLOCK,NWA,NTB,NSCH,NEQ,
     1             MI)
C***      CALL TTIME(TIM3)
C
C     CHECK FOR ZERO EIGENVALUE(S)
      NN=NEQ - ((NBLOCK-1)*NEQB)
      IF (A(NN).GT.ANORM) GO TO 10
      WRITE (6,1120)
      STOP
C
   10 TIM1=TIM2-TIM1
      TIM2=TIM3-TIM2
      IF (IFPR.EQ.1)
     * WRITE (6,1010) TIM1
      IF (IFPR.EQ.1)
     * WRITE (6,1000) TIM2
C
C   PERFORM SUBSPACE ITERN
      DO 100 I=1,NV
  100 A(I)=0.0
      NITE=0
  200 NITE=NITE+1
      WRITE (6,1020) NITE
C***      CALL TTIME(TIM1)
      N1=1+2*NV
      N2=N1+NWA
      N3=N2+NWV
      N4=N3+NWVV
      CALL REDBAK  (A(N1),A(N2),A(N3),A(N4),NEQB,NV,NWA,NWV,NWVV,NTB,
     1              NBLOCK,MI,MBAND)
C
C   SOLVE SUBSPACE EIGENVALUE PROBLEM
      N2=1+NV
      N3=N2+NV
      N4=N3+NV*NV
      N5=N4+NV*NV
      N6=N5+NV*NV
      N7=N6+NWV
```

```
      N8=N7+NWV
      N9=N8+NV
C***     CALL TTIME(TIM2)
      CALL EIGSOL (A(1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),A(N9)
     1,NF,NV,NBLOCK,NEQB,NITE,IFPR,NITEM,RTOL,IFSS,COFQ)
C***     CALL TTIME(TIM3)
      TIM1=TIM3-TIM1
      TIM2=TIM3-TIM2
      IF (IFPR.EQ.1)
     * WRITE (6,1030) TIM1
      IF (IFPR.EQ.1)
     * WRITE (6,1040) TIM2
C
      IF (NITE.LT.NITEM) GO TO 200
C
      WRITE (6,1050)
      WRITE (6,1060) (A(I),I=1,NF)
      P12=8.0D0*ATAN(1.0D0)
      DO 340 I=1,NF
      AT(I+NF)=A(I+NV)
  340 AT(I)=P12/SQRT(A(I))
C
      IF (IFSS.EQ.1) GO TO 600
C
C   APPLY STURM SEQUENCE CHECK
C***     CALL TTIME(TIM1)
      N2=1+NV
      N3=N2+NV
      N4=N3+NWA
      N5=N4+NEQB
      N6=N5+NV
      N7=N6+NV
      N8=N7+NV
      CALL SCHECK (A(1),A(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),NWA,
     1NEQB,NBLOCK,NF,NV,SHIFT,NEI,IFPR,RTOL)
      WRITE (6,1085) SHIFT
      N2=1+NWA
      N3=N2+NWA
      CALL DECOMP (A(1),A(N2),A(N3),NEQB,MBAND,NBLOCK,NWA,NTB,NSCH,NEQ,
     1            MI)
      IF (NSCH.EQ.NEI) GO TO 500
      NSCH=NSCH-NEI
      WRITE (6,1090) NSCH
      GO TO 540
  500 WRITE (6,1100) NSCH
C***  540 CALL TTIME(TIM2) !540 IS TRANSFERED TO THE NEXT LINE
540   TIM2=TIM2-TIM1
      WRITE (6,1110) TIM2
C
 600  RETURN
C
 1000 FORMAT (34HOTIME FOR STIFFNESS FACTORIZATION  F6.2)
 1010 FORMAT (42HOTIME FOR GENERATION OF INITIAL TR-VECTORS  F6.2)
 1020 FORMAT (//// 31H ITERATION NUMBER (*SSPCEB*) =   I4 //// 1X)
 1030 FORMAT (24HOTIME USED IN ITERN STEP  F6.2)
```

```
 1040  FORMAT (25HOTIME FOR EIGENVALUE SOLN  F6.2)
 1050  FORMAT (///  40H WE SOLVED FOR THE FOLLOWING EIGENVALUES     )
 1060  FORMAT (1H0,6E22.14)
 1085  FORMAT (1H1,22HCHECK APPLIED AT SHIFT  E22.14)
 1090  FORMAT (10HOTHERE ARE I4,21H EIGENVALUES MISSING  )
 1100  FORMAT (20HOWE FOUND THE LOWEST I4,12H EIGENVALUES  )
 1110  FORMAT (30HOTIME FOR STURM SEQUENCE CHECK  F6.2)
 1120  FORMAT (38HO***ERROR   SOLUTION STOP IN *SSPCEB*   / 12X,
     1         25HRIGID BODY MODE(S) FOUND., / 1X)
C
       END
      SUBROUTINE STEP
      REAL T,PT,DUM
C
C     CALLS?  ADDMAS,PLOAD,EMIDS,GROUND,INDLY,INTHIS,LOADV,INOUT,
C             TRIFAC,SOLSTP,SDSPLY
C     CALLED BY?  MAIN
C
C     CONTROL ROUTINE FOR THE STEP-BY-STEP INTEGRATION OF THE
C     EQUATIONS OF MOTION.  THE TIME DIFFERENCE FORMULA USED IS
C     THE *WILSON THETA ALGORITHM* WHICH IS UNCONDITIONALLY
C     STABLE FOR ANY CHOICE OF TIME STEP.
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /JUNK/  KK1,KK2,ISP1,ISP2,NSD,NSS,NBL,LAST,DUM(64),
     1               NUA(100),IFILL1(258)
      COMMON /DYN/   NT,NOT,ALFA,DT,BETA,NFN,NGM,NAT,IFILL2
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,KEQB,NUMEL,T(10)
      COMMON /SOL/   NBLOCK,NEQB,IFILL3(9)
C
      DIMENSION PT(7),IA(1)
      EQUIVALENCE (A(1),IA(1))
      COMMON /one/ A(1)
C
C     ASSEMBLE THE SYSTEM MASS MATRIX (DIAGONAL).  THE MASS MATRIX
C     DIAGONAL IS STORED IN CORE AS A VECTOR.  SAVE THE SYSTEM
C     MASS VECTOR ON TAPE3.
C
      PT(1) = T(9)
      N2=N1+6*NUMNP
      N3=N2+NEQ
      N4=N3+NEQB
      IF(N4.GT.MTOT) CALL ERROR (N4-MTOT)
C
      IF(MODEX.EQ.0)
     *CALL ADDMAS (A(N2),A(N3),NEQ,NEQB,NBLOCK)
C
C     D Y N A M I C   L O A D S
C
      IF(NFN.GE.1) GO TO 25
      WRITE (6,3010)
      STOP
   25 N3=N2+NFN*NEQ
      N4=N3+NFN*NEQ
      IF(N4.GT.MTOT)  CALL ERROR(N4-MTOT)
```

```
C
      CALL PLOAD (A(N1),A(N2),A(N3),NUMNP,NEQ,NFN)
      IF(NGM.EQ.0)  GO TO 100
C
C     ADD GROUND MOTION EFFECTS
C
      IF(MODEX.EQ.0)
     *CALL EMIDS (A(N1),A(N2),NUMNP,NEQ)
C
      N2=N1+NEQ*NFN
      N3=N2+NEQ*NFN
      N4=N3+NEQ
      N5=N4+NEQ
      IF(N5.GT.MTOT)  CALL ERROR (N5-MTOT)
C
      CALL GROUND (A(N1),A(N2),A(N3),A(N4),NEQ,NFN)
C
C     READ TIME DELAYS
C
  100 N2 = N1 + NEQ*NFN
      N3 = N2 + NEQ*NFN
      N4 = N3+ NAT
      IF(N4.GT.MTOT)  CALL ERROR (N4-MTOT)
C
      CALL INDLY (A(N1),A(N2),A(N3),NEQ,NFN,NAT,MAXD)
C
      N2=N1+NFN
      KN=2*NFN
C
C     READ TIME FUNCTIONS DESCRIBING LOAD HISTORIES
C
      CALL INTHIS (A(N1),A(N2),NFN,MXLP,KN)
C
C     ALLOCATE STORAGE FOR LOAD VECTOR CALCULATIONS
C
C     KN      = 2*NFN
C     NFN     = NUMBER OF TIME FUNCTIONS
C     MXLP    = MAXIMUM NUMBER OF POINTS DESCRIBING ANY FUNCTION
C     NEQ     = NUMBER OF RETAINED GLOBAL DEGREES OF FREEDOM
C
      N3=N2+KN*MXLP
      N4=N3+NEQ
      N5=N4+NFN*NEQ
      N6=N5+NFN*NEQ
      N7=N6+NEQ
      IF(N7.GT.MTOT) CALL ERROR (N7-MTOT)
C
      IF(MODEX.EQ.1) GO TO 120
C
C     COMPUTE THE SYSTEM LOAD VECTORS AT EACH SOLUTION TIME STEP
C     AND SAVE ON TAPE2.
C
      CALL LOADV (A(N1),A(N2),A(N3),A(N4),A(N5),A(N6),NEQ,NFN,KN)
C
C***  120 CALL TTIME (PT(2)) !120 IS TRANSFERED TO THE NEXT LINE
```

```
120   N2=N1+NEQ
C
C     R E A D   O U T P U T   R E Q U E S T S
C
      CALL INOUT (A(N1),A(N2),A(N2),NUMNP)
C
C***      CALL TTIME(PT(3))
C
C     RESTORE MASS MATRIX TO CORE THEREBY RELEASING TAPE3 FOR SCRATCH
C
      N2 = N1+NSD
      N3 = N2+NSS
      N4 = N3+NEQ
C
      IF(MODEX.EQ.1) GO TO 130
      REWIND 3
      MM = N4-1
      READ (3)  (A(K),K=N3,MM)
  130 CONTINUE
C
      K1 = NEQB*(2*MBAND+1)+MBAND+N4
      K2 = 4*NEQ+NSD+NSS+NEQB*(MBAND+1)+MBAND+N4
      K = K1
      IF(K2.GT.K1) K = K2
      IF(K.GT.MTOT)
     *CALL ERROR (K-MTOT)
C
      NTB = (MBAND-2)/NEQB +1
      IF(NTB.GE.NBLOCK) NTB = NBLOCK -1
C
C     PRINT EQUATION SIZE PARAMETERS
C
      WRITE (6,2003) NEQ,MBAND,NEQB,NBLOCK,NTB
C
C     D E C O M P O S E   S T I F F N E S S   M A T R I X
C
      MI = NEQB+MBAND-1
      NWA = NEQB*MBAND
      N5 = N4+NWA
      N6 = N5+NWA
      N7 = N6+MI
      IF(N7.GT.MTOT) CALL ERROR (N7-MTOT)
C
      IF(MODEX.EQ.1) GO TO 170
C
      CALL TRIFAC (A(N4),A(N5),A(N6),NEQB,MBAND,NBLOCK,NWA,NTB,NEQ,MI)
C
C***  170 CALL TTIME(PT(4)) !170 IS TRANSFERED TO THE NEXT LINE
C
C     SET UP STORAGE FOR THE TIME MARCHING SOLUTION
C
170   N5 = N4+NEQ
      N6 = N5+NEQ
      N7 = N6+NEQ
      N8 = N7+NEQ
```

```
      N9 = N8+NWA
      N10= N9+MI
C
C        1. CHECK THE AMOUNT OF REMAINING STORAGE TO SEE IF MORE THAN
C           ONE ROW CAN BE ALLOCATED TO THE ARRAYS *SDIS* AND *SSTR*.
C
      MM = MTOT-N10
      NN = NSD+NSS
      IF(NN.GT.MM) CALL ERROR (NN-MM)
      MM = MM/NN
C
C        2. COMPUTE THE NUMBER OF TIMES AT WHICH OUTPUT IS TO BE
C           PRODUCED
C
      NPT = NT/NOT
      IF(MM.GT.NPT) MM=NPT
      N11= N10+MM*NSD
C
      IF(MODEX.EQ.1) GO TO 180
C
C     S O L V E   E Q U A T I O N S   O F   M O T I O N
C
      CALL SOLSTP (IA(N1),IA(N2),A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),
     1              A(N9),A(N10),A(N11),NSD,NSS,NEQ,NEQB,MBAND,NWA,MI,
     2              MM,NBLOCK)
C
C***  180 CALL TTIME (PT(5)) !180 IS TRANSFERED TO THE NEXT LINE
180   REWIND 9
C
      IF(MODEX.EQ.1) GO TO 350
C
C     CONVERT TIME INTERVAL TO OUTPUT TIME INTERVAL
C
      DT=DFLOAT(NOT)*DT
C
C     PASS IF PRINT INTERVAL EXCEEDS THE NUMBER OF SOLUTION STEPS.
C.
      IF(NPT.LT.1) GO TO 350
C
C     COMPUTE THE NUMBER OF DISPLACEMENT RECORDS SAVED PREVIOUSLY DURING
C     THE TIME MARCHING PHASE.  EACH RECORD HAS *MM* OUTPUT VECTORS.
C
      NBL = (NPT-1)/MM +1
C
C     ALLOCATE STORAGE FOR DISPLACEMENT COMPONENT OUTPUT
C
      IF(NSD.LT.1) GO TO 350
C
C        1. NUMBER OF OUTPUT SETS AT EIGHT (8) COMPONENTS PER SET
C
      NUM = (NSD-1)/8 +1
C
C        2. COMPUTE THE NUMBER OF OUTPUT DISPLACEMENT VECTORS (AT *NSD*
C           ELEMENTS PER VECTOR) WHICH WILL FIT IN REMAINING CORE.
C           PASS IF ALL VECTORS CURRENTLY FIT IN CORE.
```

```
C
      IF(NBL.EQ.1) GO TO 270
      N2 = N1+MM*NSD
      MREM = MTOT-N2
      MMX  = MREM/NSD
C
C        3. COMPUTE THE LARGEST NUMBER OF OUTPUT VECTORS (EVENLY DIVIS-
C           IBLE BY *MM*) WHICH CAN BE RETAINED IN REMAINING CORE.  IF
C           THIS NUMBER IS AT LEAST TWICE THE EXISTING NUMBER PER RECORD
C           (*MM*), THEN ALLOW COMPACTION BEFORE OUTPUT -- OTHERWISE
C           PASS.
C
      MMX = MMX/MM
      MMX = MMX*MM
      K   = NBL*MM
      IF(MMX.GT.K) MMX = K
      NK  = 2*MM
      IF(MMX.GE.NK) GO TO 300
C
C        4. NO TAPE COMPACTIONS.
C
  270 CONTINUE
      N2 = N1
      MMX= MM
C
C     O U T P U T   S E L E C T E D   D I S P L A C E M E N T S
C
  300 CALL SDSPLY (A(N1),A(N2),MMX,MM,NSD,NUM,1,KK1,2,ISP1,NPT,4)
C
C***  350 CALL TTIME (PT(6)) !350 IS TRANSFERED TO THE NEXT LINE
C
350   IF(MODEX.EQ.1) GO TO 450
C
C     ALLOCATE STORAGE FOR ELEMENT STRESS COMPONENTS OUTPUT
C
      IF(NPT.LT.1) GO TO 450
      IF(NSS.LT.1) GO TO 450
      IF(NBL.EQ.1) GO TO 370
C
      N2 = N1+MM*NSS
      MREM = MTOT-N2
      MMX  = MREM/NSS
      MMX  = MMX/MM
      MMX  = MMX*MM
      K    = NBL*MM
      IF(MMX.GT.K) MMX = K
      NK   = 2*MM
      IF(MMX.GT.NK) GO TO 400
C
  370 CONTINUE
      N2 = N1
      MMX= MM
C
C     O U T P U T   S E L E C T E D   S T R E S S E S
C
```

```
  400 CALL SDSPLY (A(N1),A(N2),MMX,MM,NSS,NUA,NELTYP,KK2,1,ISP2,NPT,7)
C
C***  450 CALL TTIME (PT(7)) !450 IS TRANSFERED TO THE NEXT LINE
C
C     COMPUTE TIME LOG
C
450   DUM(1) = 0.0
      DO 500 I=1,6
      PT(I) = PT(I+1)-PT(I)
  500 DUM(1) = DUM(1)+PT(I)
      PT(7) = DUM(1)
      WRITE (6,2001) PT
C
C     F O R M A T S
C
 2001 FORMAT (41H1T I M E   L O G    (PARTICULAR SOLUTION), //
     1  5X,29HFORM DYNAMIC LOADS           =,F9.2 /
     2  5X,29HPROCESS OUTPUT REQUESTS      =,F9.2 /
     4  5X,29HMATRIX DECOMPOSITION         =,F9.2 /
     5  5X,29HSTEP-BY-STEP INTEGRATION     =,F9.2 /
     6  5X,29HDISPLACEMENT OUTPUT          =,F9.2 /
     7  5X,29HELEMENT STRESS OUTPUT        =,F9.2 //
     8  5X,29HTOTAL STEP-BY-STEP ANALYSIS =,F9.2 //// 1X)
 2003 FORMAT (38H1E Q U A T I O N   P A R A M E T E R S, //
     1        5X,33HTOTAL NUMBER OF EQUATIONS       =, I5 /
     2        5X,33H1/2 EQUATION BANDWIDTH          =, I5 /
     3        5X,33HNUMBER OF EQUATIONS PER BLOCK   =, I5 /
     4        5X,33HTOTAL NUMBER OF EQUATION BLOCKS =, I5 /
     5        5X,33HNUMBER OF COUPLING BLOCKS       =, I5 // 1X)
 3010 FORMAT (42H0*** ERROR   NO DYNAMIC FUNCTIONS (INPUTS), / 1X)
C
      RETURN
      END
      SUBROUTINE STRESR (SF,FI,SRM,NF,NSB,NEQB,NBLOCK)
C
C     CALLS?  ELOUTR
C     CALLED BY?  RESPEC
C
C     COMPUTE AND PRINT RMS STRESSES
C
      DIMENSION SF(12,NF),FI(NSB,NF),SRM(12)
      COMMON /EM/ SA(42,63),ND,NS,LM(63),IS(13)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
C
      DATA            HI/1HI/,HC/1HC/,HJ/1HJ/
C
C     ASSEMBLE MODESHAPES IN CORE
C
      IF(MODEX.EQ.1) RETURN
      REWIND 2
      READ (2)
      NE=NSB
      NS=NE+1-NEQB
      DO 100 I=1,NBLOCK
```

```
      READ (2) ((FI(J,K),J=NS,NE),K=1,NF)
      NS=NS-NEQB
  100 NE=NE-NEQB
  110 CONTINUE
C
      WRITE (6,2000)
      REWIND 1
      DO 500 N=1,NELTYP
      READ (1) NPAR
      NUME=NPAR(2)
C
C     CONSIDER EACH ELEMENT OF THIS TYPE  (NPAR(1))
C
      DO 400 M=1,NUME
      NEL = M
      READ (1) ND,NS,(LM(I),I=1,ND),((SA(I,J),I=1,NS),J=1,ND)
C
      NSET = (NS-1)/12 +1
      DO 390 K3=1,NSET
      K1 = (K3-1)*12 +1
      K2 = K1+11
      IF(K2.GT.NS) K2=NS
      L = 0
      DO 132 K=K1,K2
      L = L+1
  132 IS(L) = K
      IS(L+1) = 0
      L=0
C
C     COMPUTE MODAL STRESSES
C
      DO 300 I=1,12
      IF(IS(I).EQ.0) GO TO 350
      II = IS(I)
      L = L+1
      DO 200 K=1,NF
      SS = 0.0
      DO 150 J=1,ND
      JJ = LM(J)
      IF(JJ.LE.0) GO TO 150
      SS = SS + SA(II,J)* FI(JJ,K)
  150 CONTINUE
  200 SF(L,K) = SS
  300 CONTINUE
C
  350 DO 220 I=1,L
      SRM(I)=0.
      DO 220 K=1,NF
  220 SRM(I)=SRM(I)+SF(I,K)*SF(I,K)
      DO 210 I=1,L
  210 SRM(I)=SQRT(SRM(I))
C
      CALL ELOUTR (NEL,IS,L,NPAR(1),NS)
C
      WRITE (6,2030) (SRM(I),I=1,L)
```

```
      WRITE (6,2035)
C
C     IF REQUESTED, PUNCH PIPE ELEMENT (NPAR(1).EQ.12) MEMBER END FORCES
C     AND MOMENTS AT POINTS (I,J) FOR A TANGENT AND (C,J) FOR A BEND
C     A VALUE OF ONE (1) FOR NPAR(13) ACTIVATES THE PUNCH OPTION
C     NPAR(14) IS A 5 DIGIT IDENTIFIER PUNCHED IN CC 76-80 OF ALL CARDS
C
      IF(NPAR(1).NE.12) GO TO 333
      IF(NPAR(13).NE.1) GO TO 333
      IF(NS.EQ.18) GO TO 328
      WRITE (11,5000) M,(SRM(I),I=1, 6),HI,NPAR(13),NPAR(14)
      WRITE (11,5000) M,(SRM(I),I=7,12),HJ,NPAR(13),NPAR(14)
      GO TO 333
  328 IF(K3.EQ.1)
     *WRITE (11,5000) M,(SRM(I),I=7,12),HC,NPAR(13),NPAR(14)
      IF(K3.EQ.2)
     *WRITE (11,5000) M,(SRM(I),I=1, 6),HJ,NPAR(13),NPAR(14)
  333 CONTINUE
C
  390 CONTINUE
 400  CONTINUE
C
 500  CONTINUE
C
 2000 FORMAT (1H1,47HR E S P O N S E   S P E C T R U M   S T R E S S,
     1          3X,19HC O M P O N E N T S, // 23H SQUARE ROOT OF THE SUM,
     2             37H OF THE SQUARES OF THE MODAL STRESSES, /
     3             19H (FOR ALL ELEMENTS), /// 1X)
 2030 FORMAT (12E11.4)
 2035 FORMAT (1H0)
C
 5000 FORMAT (3X,I3,4X,6E10.3,A1,I2,2X,I5)
C
      RETURN
      END
      SUBROUTINE STRETR (NNS,RHOM)
C
C     CALLS?  QDCOS,TDCOS,TRFPRD,CSTSTR,LCT9ST
C     CALLED BY?  TPLATE
C
C     THIS ROUTINE FORMS THE ELEMENT*S MASS MATRIX AND THE
C     STRESS (MOMENT) - DISPLACEMENT TRANSFORMATION MATRIX
C     EVALUATED AT THE ELEMENT CENTROID
C     THE GLOBAL FORCES DUE TO A UNIT VALUE OF APPLIED NORMAL PRESSURE
C     ARE ALSO CALCULATED
C
      COMMON /QTSARG/
     1 XX(5),YY(5),ZZ(5),HM(5),HP(5),CM(3,3),ALFA(3),HW(5),RHO(5,3),
     2 P(5),T(5),DT(5),SM(5,3),BM(5,3),TDIS(36),TROT(36),S(30,30),
     3 R(30)
      COMMON /TRIARG/
     1 A(3),B(3),H(3),HPT(3),C(3,3),SMT(3,3),BMT(3,3),FT(12),
     2 PX(3),PY(3),PT(3),RM(3),ST(12,12)
      COMMON /EM/
     1 LM(24),ND,NS,STRAN(6,30),NC(3),IPAD,AREA,XMM,TD1(13),TD2(13),
```

```
     2 TD3(9),TR1(9),TR2(9),TR3(9),SCST(3,6),XST(3,6),SLCT9(3,9),
     3 XLCT9(3,9),DUMMY(238),RF(24,4),XM(24),SA(12,24),SF(12,4),PF(24),
     4  IFILL(3000)
      COMMON /TRANSF/
     1 T1(3),T2(3),T3(3),T0(3,3)
C
      DIMENSION IPERMQ(4)
C
      DATA      IPERMQ/2,3,4,1/
C
      NEN = MIN0(NNS,4)
      WG = 1.0
      IF(NEN.EQ.4) WG = 0.25
      N3 = 2*NEN - 3
      NC(3)  = N3
      NTRI  = 3*NEN - 8
C
C     COMPUTE DIRECTION COSINE ARRAY FOR THE ELEMENT AXES
C
      CALL QDCOS  (NTRI,XX,YY,ZZ,T0)
C
C     CLEAR THE MASS MATRIX VECTOR, STRESS TRANSFORMATION ARRAY AND THE
C     GLOBAL FORCE VECTOR DUE TO A UNIT NORMAL PRESSURE
C
      DO 10 K=1,ND
      XM(K)  = 0.0
      PF(K)  = 0.0
      DO 5 I=1,NS
    5 STRAN(I,K)  = 0.0
   10 CONTINUE
      IF(NTRI.EQ.1) GO TO 20
      DO 15 I=25,30
      DO 15 J=1,NS
   15 STRAN(J,I)  = 0.0
   20 CONTINUE
C
C     LOOP OVER  NTRI  TRIANGLE SUB-ELEMENTS ASSEMBLING STRESS/
C     DISPLACEMENT TRANSFORMATION AND MASS MATRICES
C
      DO 300 NT=1,NTRI
      N1 = NT
      N2 = IPERMQ(N1)
      NC(1)  = N1
      NC(2)  = N2
C
C     COMPUTE DIRECTION COSINES OF LOCAL TRIANGLE SYSTEM AND THE
C     TRIANGLE PROJECTIONS (A,B) ONTO THE LOCAL X,Y PLANE
C
      CALL TDCOS  (N1,N2,N3,XX,YY,ZZ,A,B)
C
C     COMPUTE MASS COEFFICIENTS FOR THE SUB-ELEMENT (TRIANGLE) AND
C     ASSEMBLE INTO THE MASS ARRAY.  ALSO, COMPUTE NODAL FORCES
C     DUE TO UNIT VALUE OF NORMAL PRESSURE.
C
      AREA = (A(3)*B(2)  - A(2)*B(3))* 0.5
```

```
      XMM = (HW(N1)+HW(N2)+HW(N3))* AREA* RHOM/ 9.0
      FAC = AREA/ 3.0
C
      DO 40 I=1,2
      K = 6*(NC(I)-1)
      DO 30 L=1,3
      K = K+1
      PF(K) = PF(K) + FAC* T3(L)
   30 XM(K) = XM(K) + XMM
   40 CONTINUE
      DUM = XMM* 0.5
      DU2 = FAC* 0.5
      IF(NTRI.EQ.1) GO TO 45
      K1 = 6*(N1-1)
      K2 = 6*(N2-1)
      GO TO 50
   45 K1 = 6*(N3-1)
      K2 = K1
   50 DO 60 L=1,3
      K1 = K1+1
      K2 = K2+1
      PF(K1) = PF(K1) + DU2* T3(L)
      XM(K1) = XM(K1) + DUM
      PF(K2) = PF(K2) + DU2* T3(L)
   60 XM(K2) = XM(K2) + DUM
C
C     FORM TRANSFORMATIONS BETWEEN SUB-ELEMENT (TRIANGLE) LOCAL
C     SYSTEM AND THE GLOBAL COORDINATE SYSTEM
C
      CALL TRFPRD (0,NEN,TDIS,TDIS,TDIS,TD1,TD2,TD3)
      CALL TRFPRD (0,NEN,TROT,TROT,TROT,TR1,TR2,TR3)
C
C     MEMBRANE CONTRIBUTION
C
      CALL CSTSTR (SCST,XST)
C
      K1 = 0
      DO 100 JJ=1,3
      M = 6*(NC(JJ)-1)
      DO 100 L=1,3
      M = M+1
      K1 = K1+1
      DO 80 K2=1,3
      STRAN(K2,M) = STRAN(K2,M) +(SCST(K2,JJ  )* TD1(K1)
     1                          + SCST(K2,JJ+3)* TD2(K1))* WG
   80 CONTINUE
  100 CONTINUE
C
C     BENDING CONTRIBUTION
C
      DO 110 K=1,3
      N = NC(K)
  110 HPT(K) = HP(N)
C
      CALL LCT9ST (SLCT9,3,XLCT9)
```

```
C
      DO 200 JJ=1,3
      I = 3*(JJ-1)
      M = 6*(NC(JJ)-1)
      DO 180 L=1,6
      M = M+1
      IF(L.GT.3) GO TO 120
      K1 = I+L
      DO 115 K2=1,3
  115 STRAN(K2+3,M) = STRAN(K2+3,M) + SLCT9(K2,JJ)* TD3(K1)* WG
      GO TO 180
  120 K1 = I+L-3
      DO 130 K2=1,3
      STRAN(K2+3,M) = STRAN(K2+3,M) + (SLCT9(K2,JJ+3)* TR1(K1)
     1                               + SLCT9(K2,JJ+6)* TR2(K1))* WG
  130 CONTINUE
  180 CONTINUE
  200 CONTINUE
C
  300 CONTINUE
C
C     PERFORM CONDENSATION ON INTERNAL DEGREES OF FREEDOM FOR
C     QUADRILATERAL ELEMENT*S STRESS/DISPLACEMENT TRANSFORMATION
C
      IF(NTRI.EQ.1) GO TO 500
C
      DO 400 N=1,6
      K = 30-N
      L = K+1
      PIV = S(L,L)
      IF(PIV.LT.1.OE-8) GO TO 400
      DO 390 K1=1,6
      DUM = STRAN(K1,L)
      STRAN(K1,L) = STRAN(K1,L)/ PIV
      DO 380 I=1,K
      STRAN(K1,I) = STRAN(K1,I) - S(I,L)* DUM
  380 CONTINUE
  390 CONTINUE
  400 CONTINUE
C
  500 DO 510 K1=1,NS
      DO 510 K2=1,ND
      SA(K1,K2) = STRAN(K1,K2)
  510 CONTINUE
C
      RETURN
      END
      SUBROUTINE STRSD1 (NUM,SF,FI,X,NF,NSB,NDS,NEQB,NBLOCK)
C
C     CALLS?  DISPLY
C     CALLED BY?  HISTRY
C
      DIMENSION NUM(1),SF(8,NF),FI(NSB,NF),X(NF,NDS)
      COMMON /EM/ ND,NS,LM(100),SA(1),IFILL2(5034)
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
```

```
      COMMON /JUNK/ N,NEL,IS(13),M,I,L,KS(3,8),II,K,SS,J,JJ
     .                ,NUME,NE,IFILL1(380)
      COMMON /EXTRA/ MODEX,NT8,IFILL3(14)
C
C     ASSEMBLE MODE SHAPES IN CORE
C
      IF(MODEX.EQ.1) GO TO 110
      REWIND 7
      READ (7)
      NE=NSB
      NS=NE+1-NEQB
      DO 100 I=1,NBLOCK
      READ (7)  ((FI(J,K),J=NS,NE),K=1,NF)
      NS=NS-NEQB
  100 NE=NE-NEQB
C
C     FORM STRESS MATRIX,MODE SHAPE TRANSFORMATION FOR
C     SELECTED STRESS COMPONENTS ONLY.
C
      REWIND 1
      REWIND 3
C
  110 CONTINUE
      READ (5,1000) KKK,ISP
      WRITE (6,3000)
      IF(MODEX.EQ.1) GO TO 600
      DO 500 N=1,NELTYP
      READ (1) NPAR
      WRITE (3) NPAR
      WRITE (6,3001) NPAR(1)
      READ (5,1000) NEL,IS
      WRITE (6,2000) NEL,(IS(I),I=1,12)
      NUME=NPAR(2)
      L=0
      NUM(N)=0
C
      DO 400 M=1,NUME
      IF (NEL.EQ.M) GO TO 410
      READ (1)
      GO TO 400
 410  READ (1) ND,NS
      BACKSPACE 1
      NDT = NS* ND
      READ (1) ND,NS,(LM(I),I=1,ND),(SA(K),K=1,NDT)
C
      DO 300 I=1,NS
      II=IS(I)
      IF(II.EQ.0) GO TO 350
      L=L+1
      KS(1,L)=NEL
      KS(2,L)=II
C
      DO 200 K=1,NF
      SS=0.
      KK = II
```

```
      DO 150 J=1,ND
      JJ=LM(J)
      IF(JJ) 150,150,140
  140 SS=SS+SA(KK)*FI(JJ,K)
  150 KK=KK+NS
  200 SF(L,K)=SS
C
      IF(L.LT.8) GO TO 300
      WRITE (3) L,KS,SF,NS
      L=0
      NUM(N)=NUM(N) + 1
  300 CONTINUE
  350 READ  (5,1000) NEL,IS
      WRITE (6,2000) NEL,(IS(I),I=1,12)
  400 CONTINUE
C
      IF(L.EQ.0) GO TO 500
      WRITE (3) L,KS,SF,NS
      NUM(N)=NUM(N) + 1
  500 CONTINUE
      WRITE (6,4000) KKK,ISP
C
C     COMPUTE AND OUTPUT HISTORY OF VALUES
C
      CALL DISPLY (X,SF,NF,NDS,NUM,NELTYP,KKK,1,ISP)
C
      RETURN
C
C     READ OUTPUT REQUESTS FOR THE DIFFERENT ELEMENTS
C
  600 L = 0
  610 L=L + 1
      WRITE (6,3010) L
 3010 FORMAT (/// 36H OUTPUT REQUESTS FOR ELEMENT GROUP =,I3,//
     1              8H ELEMENT,5X,25HDESIRED STRESS COMPONENTS  )
  630 READ (5,1000) NEL,IS
      IF (NEL.LT.1) GO TO 620
      WRITE (6,2000) NEL,(IS(I),I=1,12)
      GO TO 630
  620 IF (L.LT.NELTYP) GO TO 610
      WRITE (6,4000) KKK,ISP
      RETURN
C
 1000 FORMAT (14I5)
 2000 FORMAT(4X,I4,3X,12I3)
 3000 FORMAT (25H1ELEMENT STRESS COMPONENT, /
     1          22H TIME HISTORY REQUESTS,    // 1X)
 3001 FORMAT (22H ELEMENT TYPE NUMBER =, I3 // 8H ELEMENT,5X,
     1 11HS T R E S S,6X,17HC O M P O N E N T,/ 8H  NUMBER,3X,
     2 12(3H  *) / 1X)
 4000 FORMAT (// 25H CODE FOR OUTPUT TYPE   =, I2 /
     1           3X,19HEQ.1, HISTORY TABLE,     /
     2           3X,18HEQ.2, PRINTER PLOT,      /
     3           3X,17HEQ.3, MAXIMA ONLY,       /
     4           25H PRINTER PLOT SPACING   =, I2 / 1X)
```

```
C
      END
      SUBROUTINE ST8R21  (E,B,S,XX,NOD9,H,P,SIGDT,DELT,FT,DL,XM,NEL,ND,
     1                    IELD,IELX,KTL,KGL,KMS,NINT,NINTZ,WTDEN,MSDEN)
C
C     CALLED BY ? THDFE
C     CALLS ? DER3DS
C
C
C
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C .                                                                   .
C .                                                                   .
C .      HEXAHEDRAL CURVILINEAR THREE-DIMENSIONAL ELEMENTS            .
C .                                                                   .
C .      ISOPARAMETRIC OR SUBPARAMETRIC                               .
C .                                                                   .
C .                                                                   .
C . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
C
C
C
      DIMENSION E(6,1),B(6,1),S(63,1),XX(3,1),NOD9(1),H(1),P(3,1),
     1          SIGDT(1),DELT(1),FT(1),DL(1),XM(1),D(9),SDT(6),BV(63),
     2          W(3,3),IPERM(3,3),KDX(3),LDX(3)
C
      COMMON /GAUSS/ XG(4,4),WGT(4,4)
C     REAL MSDEN
      REAL MSDEN
C
      DATA IPERM / 1,4,6, 4,2,5, 6,5,3 /
C
      VOL = 0.0
C
C     DETERMINE IF THE MATERIAL IS ORTHOTROPIC (ISO.EQ.1, ISOTROPIC)
C
      DUM = 0.0
      DO 20 I=4,6
      J = I-1
      DO 20 K=1,J
   20 DUM = DUM +ABS(E(K,I))
      ISO = 1
      IF(DUM.GT.1.0E-6) ISO = 0
      IF(ISO.EQ.0) GO TO 24
      DO 22 I=2,3
      DUM = DUM +ABS(E(I  ,I  ) -E(I-1,I-1))
   22 DUM = DUM +ABS(E(I+3,I+3) -E(I+2,I+2))
      DUM = DUM +ABS(E(1  ,2 ) - E(2  ,3  ))
      DUM = DUM +ABS(E(2  ,3  ) - E(3  ,1  ))
      IF ( DUM.GT.1.0E-6 ) ISO=0
   24 CONTINUE
C
C
C     VOLUME INTEGRATION LOOP
C
```

```
C
      DO 10 LX=1,NINT
      DO 10 LY=1,NINT
      E1=XG(LX,NINT)
      E2=XG(LY,NINT)
      DO 10 LZ=1,NINTZ
      E3=XG(LZ,NINTZ)
C
      WT=WGT(LX,NINT)*WGT(LY,NINT)*WGT(LZ,NINTZ)
C
C     EVALUATE STRAIN-DISPLACEMENT MATRIX B AND JACOBIAN DETERMINANT
C
      CALL DER3DS (NEL,XX,B,DET,E1,E2,E3,NOD9,H,P,IELD,IELX)
C
C     ADD CONTRIBUTION TO ELEMENT STIFFNESS
C
      FACT = WT* DET
      FACT2 =SQRT(FACT)
C
      DO 25 I=1,IELD
      K3 = 3*I
      K2 = K3-1
      K1 = K2-1
      BV(K1) = B(1,K1)* FACT2
      BV(K2) = B(2,K2)* FACT2
      BV(K3) = B(3,K3)* FACT2
   25 CONTINUE
C
      DO 30 I=1,ND
      DO 30 J=1,ND
   30 S(I,J) = S(I,J) + BV(I)* BV(J)
C
C     ACCUMULATE ELEMENT VOLUME
C
      VOL = VOL + FACT
C
C     COMPUTE GRAVITY LOADS
C
      IF(KGL.EQ.0) GO TO 150
      DO 130 K=1,IELD
  130 DL(K) = DL(K) + H(K)*FACT* WTDEN
C
C -   COMPUTE THERMAL LOADING NODE FORCE VECTOR
C
  150 IF(KTL.EQ.0) GO TO 190
C
C        1. ELEMENT TEMPERATURE DIFFERENCE AT THIS INTEGRATION POINT
C           (R,S,T)
C
      DT = 0.0
      DO 160 K=1,IELD
  160 DT = DT + H(K)* DELT(K)
      DT = DT* FACT
C
C        2. INITIAL STRESSES AT (R,S,T)
```

```
C
      DO 170 K=1,6
  170 SDT(K) = SIGDT(K)*DT
C
C        3. NODE FORCES
C
      DO 180 K=1,ND
      DO 175 I=1,6
  175 FT(K) = FT(K) + B(I,K)* SDT(I)
  180 CONTINUE
C
  190 CONTINUE
   10 CONTINUE
C
      DO 35 I=1,2
      IC = ND-1
      DO 35 J=1,IC
      M=J+1
   35 S(M,J) = S(J,M)
C
C     COMPLETE THE K-MATRIX WITH APPROPRIATE MATERIAL CONSTANT MULT-
C     PLICATIONS OF THE INTEGRATED B(I)*B(J) ARRAY.
C
C        1. TEST FOR MATERIAL TYPE
C
      IF(ISO.EQ.0) GO TO 75
C
C           A.  ISOTROPIC MATERIAL
C
      D1 = E(1,1)
      D2 = E(1,2)
      D3 = E(4,4)
C
      DO 60 I=1,IELD
      K3 = 3*I
      K2 = K3-1
      K1 = K2-1
      K0 = K1-1
      DO 60 J=1,IELD
      L3 = 3*J
      L2 = L3-1
      L1 = L2-1
      L0 = L1-1
C
      IC = 0
      DO 40II=1,3
      M = II+ K0
      DO 40 JJ=1,3
      N = JJ+ L0
      IC = IC+ 1
      D(IC) = S(M,N)
   40 CONTINUE
C
      S(K1,L1) = D(1)* D1 + (D(5) + D(9))* D3
      S(K2,L2) = D(5)* D1 + (D(1) + D(9))* D3
```

```
      S(K3,L3) = D(9)* D1 + (D(5) + D(1))* D3
      S(K1,L2) = D(2)* D2 + D(4)* D3
      S(K2,L1) = D(4)* D2 + D(2)* D3
      S(K2,L3) = D(6)* D2 + D(8)* D3
      S(K3,L2) = D(8)* D2 + D(6)* D3
      S(K3,L1) = D(7)* D2 + D(3)* D3
      S(K1,L3) = D(3)* D2 + D(7)* D3
C
   60 CONTINUE
C
      GO TO 110
C
C           B. ANISOTROPIC MATERIAL
C
   75 DO 100 I=1,IELD
      KO = 3*I-3
      DO 100 J=1,IELD
      LO = 3*J-3
C
      DO 80 II=1,3
      M = II+KO
      DO 80 JJ=1,3
      N = JJ+LO
      W(II,JJ) = S(M,N)
   80 CONTINUE
C
      DO 100 K=1,3
      I1 = KO+K
      DO 82 IJ=1,3
   82 KDX(IJ)=IPERM(IJ,K)
      DO 95 L=1,3
      I2 = LO+L
      DO 83 IJ=1,3
   83 LDX(IJ)=IPERM(IJ,L)
C
      SUM=0.0
C
      DO 90 II=1,3
      K1 = KDX(II)
      DO 85 JJ=1,3
      K2 = LDX(JJ)
C
   85 SUM = SUM + W(II,JJ)*E(K1,K2)
   90 CONTINUE
C
      S(I1,I2) = SUM
C
   95 CONTINUE
  100 CONTINUE
  110 CONTINUE
C
C
C     REFLECT FOR SYMMETRY
C
      DO 200 I=1,ND
```

```
      DO 200 J=I,ND
  200 S(J,I) = S(I,J)
C
C     CONSTRUCT THE LUMPED MASS MATRIX
C
      IF(KMS.EQ.0) RETURN
C
      FACT = VOL* MSDEN/ IELD
      DO 220 K=1,ND
  220 XM(K) = FACT
C
C
      RETURN
      END
      SUBROUTINE TANGDC (NEL,X1,X2,ACARD,A,MODEX,XLN)
C
C     CALLED BY?  PIPEK
C
C     COMPUTATION OF DIRECTION COSINE ARRAY FOR THE LOCAL AXES OF PIPE
C     TANGENT ELEMENT
C
C     NEL            =  ELEMENT NUMBER
C     X1             =  GLOBAL COORDINATES OF END I
C     X2             =  GLOBAL COORDINATES OF END J
C     ACARD          =  GLOBAL PROJECTIONS OF THE LOCAL Y-AXIS AS INPUT
C                       ON THE ELEMENT CARD
C     A              =  MATRIX OF DIRECTION COSINES RELATING LOCAL TO THE
C                       GLOBAL SYSTEM.  A(I,J) IS THE PROJECTION ON THE
C                       I-TH GLOBAL AXIS OF A UNIT VECTOR IN THE LOCAL
C                       J-DIRECTION.
C     MODEX          =  EXECUTION MODE
C                       (EQ.0, SOLUTION)
C                       (EQ.1, DATA CHECK)
C     XLN            =  TANGENT ELEMENT LENGTH
C
      DIMENSION X1(3),X2(3),ACARD(3),A(3,3)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
C     LOCAL X-AXIS FROM NODE I TO NODE J
C
      A(1,1) = X2(1)-X1(1)
      A(2,1) = X2(2)-X1(2)
      A(3,1) = X2(3)-X1(3)
      XLN = A(1,1)**2 + A(2,1)**2 + A(3,1)**2
      XLN =SQRT(XLN)
      IF(XLN.GT.1.0E-8) GO TO 20
      WRITE (6,3000) NEL
      MODEX = 1
      RETURN
   20 DUM = 1.0/XLN
      DO 25 K=1,3
   25 A(K,1) = A(K,1)* DUM
C
C     LOCAL Y-AXIS
```

```
C
C        1. TEST FOR USER INPUT FROM THE ELEMENT CARD
C
      DUM = ACARD(1)**2 + ACARD(2)**2 + ACARD(3)**2
      IF(DUM.LT.1.0E-8) GO TO 40
C
C        2. DIRECT USER INPUT OF THE LOCAL Y-AXIS
C
      DUM = 1.0/SQRT(DUM)
      DO 30 K=1,3
   30 A(K,2) = ACARD(K)* DUM
C
C        3. TEST FOR ZERO PROJECTION OF THE INPUT Y-AXIS ON THE KNOWN
C           LOCAL X-AXIS DIRECTION
C
      DUM = A(1,1)*A(1,2) + A(2,1)*A(2,2) + A(3,1)*A(3,2)
      DUM =ABS(DUM)
C
      IF(DUM.LT.1.0E-6) GO TO 60
      WRITE (6,3010) NEL
C
C        4. COMPUTE THE ORIENTATION OF THE Y-AXIS USING THE DEFAULT
C           CONVENTION
C
   40 DU2 = A(1,1)**2 + A(3,1)**2
C
C        5. TEST FOR FOR THE CASE OF THE MEMBER LONGITUDINAL AXIS
C           BEING PARALLEL TO THE GLOBAL Y-AXIS
C
      IF(DU2.GT.1.0E-12) GO TO 50
C
C        6. VERTICAL MEMBER
C
      A(1,2) = 0.0
      A(2,2) = 0.0
      A(3,2) = 1.0
      GO TO 60
C
C        7. NON-VERTICAL MEMBER
C
   50 DU2 =SQRT(DU2)
      A(1,2) = -A(1,1)*A(2,1)/DU2
      A(2,2) =  DU2
      A(3,2) = -A(3,1)*A(2,1)/DU2
C
C     LOCAL Z-AXIS
C
   60 CONTINUE
      A(1,3) = A(2,1)*A(3,2) - A(3,1)*A(2,2)
      A(2,3) = A(3,1)*A(1,2) - A(1,1)*A(3,2)
      A(3,3) = A(1,1)*A(2,2) - A(2,1)*A(1,2)
C
      RETURN
C
 3000 FORMAT (36HOERROR***  ZERO LENGTH FOR ELEMENT (,I4, 2H)., / 1X)
```

```
 3010 FORMAT (51HOERROR***  USER DEFINED Y-AXIS IS NOT PERPENDICULAR,
     1 46H TO THE LONGITUDINAL AXIS OF TANGENT ELEMENT (,I4,2H)., /
     2 11X,27HDEFAULT CONVENTION ASSUMED., / 1X)
C
      END
C
C     CALLS?  PINVER
C     CALLED BY?  PIPEK
C
C     COMPUTATION OF THE ELEMENT STIFFNESS AND LOAD MATRICES FOR A
C     TANGENT (STRAIGHT) PIPE ELEMENT.
C
C
C     ALFAV        =  SHAPE FACTOR FOR SHEAR DISTORTION
C                     (GT.99.99, NEGLECT)
C     E            =  YOUNG*S MODULUS
C     XNU          =  POISSON*S RATIO
C     AREA         =  SECTION AREA
C     XMI          =  MOMENT OF INERTIA
C     NODE         =  NODE NUMBER AT END J OF THE TANGENT
C     NEL          =  PIPE ELEMENT NUMBER
C     MODEX        =  EXECUTION MODE
C                     (EQ.1, DATA CHECK)
C     F(6,6)       =  FLEXIBILITY MATRIX AT NODE J
C     XLN          =  LENGTH OF THE TANGENT
C     THERM        =  THERMAL EXPANSION COEFFICIENT
C     P            =  INTERNAL PIPE PRESSURE
C     WALL         =  PIPE WALL THICKNESS
C     DOUT         =  OUTSIDE DIAMETER OF THE PIPE
C     B            =  FREE END DEFLECTIONS AT NODE J DUE TO
C                     (1)  UNIFORM LOAD IN THE X    DIRECTION
C                     (2)  UNIFORM LOAD IN THE Y    DIRECTION
C                     (3)  UNIFORM LOAD IN THE Z    DIRECTION
C                     (4)  UNIFORM THERMAL EXPANSION (DT=1)
C                     (5)  P,    INTERNAL PRESSURE
C     H            =  FORCE TRANSFORMATION RELATING REACTIONS AT NODE I
C                     DUE TO UNIT LOADS AT NODE J
C     S            =  LOCAL TANGENT ELEMENT STIFFNESS MATRIX
C     FEF          =  FIXED END FORCES (ACTING ON THE NODES) DUE TO
C                     (1)  UNIFORM LOAD IN THE X    DIRECTION
C                     (2)  UNIFORM LOAD IN THE Y    DIRECTION
C                     (3)  UNIFORM LOAD IN THE Z    DIRECTION
C                     (4)  UNIFORM THERMAL EXPANSION (DT=1)
C                     (5)  P,    INTERNAL PRESSURE
C     XM           =  LUMPED MASS MATRIX
C     SA           =  STRESS-DISPLACEMENT TRANSFORMATION RELATING THE
C                     12 GLOBAL COMPONENTS OF DISPLACEMENT TO THE 6
C                     LOCAL COMPONENTS OF MEMBER LOADS LOCATED AT NODE
C                     I AND AT NODE J.
C     FEFC         =  FIXED-END FORCE CORRECTIONS TO THE MEMBER LOADS
C                     DUE TO THE FIVE (5) TYPES OF ELEMENT LOADS
C     XMAS         =  MASS   PER UNIT LENGTH OF THE SECTION
C     DC           =  ARRAY OF DIRECTION COSINES WHICH TRANSFORMS LOCAL
C                     VECTORS TO GLOBAL VECTORS
      SUBROUTINE TANGKS
```

```
      COMMON /PIPEC/ ALFAV,E,XNU,DU1,DU2,IDUM3,NODE,NEL,MODEX,
     1               XLN,THERM,P,AREA,XMI,WALL,DOUT,XMAS
      COMMON /EM/    IXX(14),S(12,12),RF(12,4),XM(12),SA(18,12),
     1               SF(18,4),FEF(12,5),FEFC(18,5),F(6,6),B(6,6),
     2               H(6,6),DC(3,3),IFILL2(3606)
      COMMON /ELPAR/ NPAR(14),IFILL1(10)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
C     SET THE FACTOR FOR AXIAL DEFORMATIONS
C
      AXIAL = 1.0
C
C     SET THE FACTOR FOR SHEAR DEFORMATIONS (EQ.0, NEGLECT)
C
      XKAP = ALFAV
      IF(ALFAV.GT.99.99) XKAP = 0.0
C
C     COMPUTE THE MATERIAL FACTORS
C
      RE = 1.0/E
      XNU1 = 1.0+XNU
C
C     COMPUTE SECTION PROPERTY CONSTANTS
C
      RA = AXIAL*XLN*RE/AREA
      RV = 2.0*XKAP*XNU1*XLN*RE/AREA
      RT = XNU1*XLN*RE/XMI
      RB = XLN*RE/XMI
      XL2 = XLN**2
C
C     FORM THE NODE FLEXIBILITY MATRIX AT NODE J REFERENCED TO THE
C     LOCAL (X,Y,Z) COORDINATE SYSTEM AT NODE I.
C
C     X - DIRECTION...  AXIAL FROM NODE I TO NODE J
C     Y - DIRECTION...  TRANSVERSE BENDING AXIS
C     Z - DIRECTION...  TRANSVERSE BENDING AXIS ORTHOGONAL TO X AND Y
C
      DO 50 I=1,6
      DO 50 K=1,6
      F(I,K) = 0.0
   50 CONTINUE
C
C     A X I A L
C
      F(1,1) = F(1,1) + RA
C
C     S H E A R
C
      F(2,2) = F(2,2) + RV
      F(3,3) = F(3,3) + RV
C
C     T O R S I O N
C
      F(4,4) = F(4,4) + RT
```

```
C
C     B E N D I N G
C
      F(2,2) = F(2,2) + RB*XL2/3.0
      F(3,3) = F(3,3) + RB*XL2/3.0
      F(5,5) = F(5,5) + RB
      F(6,6) = F(6,6) + RB
      F(2,6) = F(2,6) + RB*XLN*0.5
      F(3,5) = F(3,5) - RB*XLN*0.5
C
      DO 60 I=1,6
      DO 60 K=1,6
      F(K,I) = F(I,K)
   60 CONTINUE
C**** PRINT THE NODE FLEXIBILITY MATRIX
      IF(NPAR(9).LT.1) GO TO 6690
      WRITE (6,4000)
      WRITE (6,4010) ((F(I,K),K=1,6),I=1,6)
 6690 CONTINUE
C****
C
C     FORM THE NODE STIFFNESS MATRIX
C
      CALL PINVER (F,6,6,NODE,NEL,MODEX)
C**** PRINT THE NODE STIFFNESS MATRIX
      IF(NPAR(9).LT.1) GO TO 6691
      WRITE (6,4020)
      WRITE (6,4030) ((F(I,K),K=1,6),I=1,6)
 6691 CONTINUE
C****
C
C     COMPUTE THE DEFLECTIONS/ROTATIONS (MEASURED IN THE X,Y,Z SYSTEM
C     AT NODE I) AT NODE J DUE TO UNIFORM LOADS IN EACH OF THE X,Y,Z
C     DIRECTIONS (AT I).  THE UNIFORM LOADS ARE DIRECTION INVARIANT
C     WITH POSITION ALONG THE LENGTH, AND NODE I IS COMPLETELY FIXED
C     WHILE NODE J IS FREE.
C
      DO 70 I=1,6
      DO 70 K=1,3
      B(I,K) = 0.0
   70 CONTINUE
C
C     A X I A L
C
      RA = 0.5*RA*XLN
      B(1,1) = B(1,1) + RA
C
C     S H E A R
C
      RV = 0.5*RV*XLN
      B(2,2) = B(2,2) + RV
      B(3,3) = B(3,3) + RV
C
C     B E N D I N G
C
```

```
      RB = RB*XL2/6.0
      B(2,2) = B(2,2) + RB*XLN*0.75
      B(3,3) = B(3,3) + RB*XLN*0.75
      B(5,3) = B(5,3) - RB
      B(6,2) = B(6,2) + RB
C
C     COMPUTE THE FREE NODE DEFLECTIONS AT END J DUE TO A UNIFORM
C     THERMAL EXPANSION
C
      DO 80 I=1,6
      B(I,4) = 0.0
   80 CONTINUE
C
      B(1,4) = XLN*THERM
C
C     COMPUTE THE FREE NODE DEFLECTIONS AT END J DUE TO PRESSURE
C
      DO 90 I=1,6
      B(I,5) = 0.0
   90 CONTINUE
C
C     MEL REPORT 10-66, EQUATION (3-28).
C
      RM = (DOUT-WALL)*0.5
      B(1,5) = 0.5*P*RM*RE/WALL*(1.0-2.0*XNU)*XLN
C**** PRINT THE FREE END DEFLECTIONS
      IF(NPAR(9).LT.1) GO TO 6692
      WRITE (6,4050)
      WRITE (6,4060) ((B(I,K),K=1,5),I=1,6)
 6692 CONTINUE
C****
C
C     SET UP THE FORCE TRANSFORMATION RELATING REACTIONS AT NODE I
C     ACTING ON THE MEMBER END DUE TO UNIT LOADS APPLIED TO THE MEMBER
C     END AT NODE J.
C
      DO 100 I=1,6
      DO 100 K=1,6
      H(I,K) = 0.0
  100 CONTINUE
C
      DO 105 K=1,6
      H(K,K) =-1.0
  105 CONTINUE
C
      H(6,2) = -XLN
      H(5,3) =  XLN
C
C     FORM THE UPPER TRIANGULAR PORTION OF THE LOCAL ELEMENT STIFFNESS
C     MATRIX FOR THE TANGENT
C
      DO 110 K=1,6
      DO 110 I=K,6
      S(K+6,I+6) = F(K,I)
  110 CONTINUE
```

```
C
      DO 130 IR=1,6
      DO 130 IC=1,6
      S(IR,IC+6) = 0.0
      DO 120 IN=1,6
      S(IR,IC+6) = S(IR,IC+6) + H(IR,IN)* F(IN,IC)
  120 CONTINUE
  130 CONTINUE
C
      DO 150 IR=1,6
      DO 150 IC=IR,6
      S(IR,IC) = 0.0
      DO 140 IN=1,6
      S(IR,IC) = S(IR,IC) + S(IR,IN+6)* H(IC,IN)
  140 CONTINUE
  150 CONTINUE
C
C     REFLECT FOR SYMMETRY
C
      DO 160 I=1,12
      DO 160 K=1,12
      S(K,I) = S(I,K)
  160 CONTINUE
C**** PRINT THE STIFFNESS MATRIX (LOCAL) FOR THE TANGENT
      IF(NPAR(9).LT.1) GO TO 6693
      WRITE (6,4500)
      WRITE (6,4510) ((S(I,J),J=1,6 ),I=1,12)
      WRITE (6,4510) ((S(I,J),J=7,12),I=1,12)
 6693 CONTINUE
C****
C
C     COMPUTE THE RESTRAINED NODE FORCES ACTING ON THE NODES OF THE
C     TANGENT DUE TO THE MEMBER LOADINGS
C
      DO 180 I=1,5
      DO 180 J=1,12
      FEF(J,I) = 0.0
      DO 170 K=1,6
      FEF(J,I) = FEF(J,I) - S(J,K+6)* B(K,I)
  170 CONTINUE
  180 CONTINUE
C
C     FOR THE DISTRIBUTED LOADS SUPERIMPOSE THE CANTILEVER REACTIONS
C     ACTING ON THE ELEMENT AT NODE I.
C
      DUM = 0.5*XL2
      FEF(1,1) = FEF(1,1) - XLN
C
      FEF(2,2) = FEF(2,2) - XLN
      FEF(6,2) = FEF(6,2) - DUM
C
      FEF(3,3) = FEF(3,3) - XLN
      FEF(5,3) = FEF(5,3) + DUM
C**** PRINT THE FIXED END QUANTITIES
      IF(NPAR(9).LT.1) GO TO 6694
```

```
      WRITE (6,4600)
      WRITE (6,4610) ((FEF(I,J),J=1,5),I=1,12)
 6694 CONTINUE
C****
C
C     FORM THE LUMPED MASS MATRIX
C
      DUM = 0.5*XLN*XMAS
      DO 200 K=1,3
      XM(K)   = DUM
      XM(K+6) = DUM
      XM(K+3) = 0.0
      XM(K+9) = 0.0
  200 CONTINUE
C
C     COMPUTE THE FIXED-NODE CORRECTIONS TO THE SECTION STRESSES
C     DUE TO ELEMENT LOADINGS.  FORCES ACT ON THE SEGMENT BETWEEN
C     THE POINT OF EVALUATION AND NODE I.
C
C        1. AT NODE I
C
      DO 210 I=1,5
      DO 210 K=1,6
      FEFC(K,I) = -FEF(K,I)
C
C        2. AT NODE J
C
      FEFC(K+6,I) = FEF(K+6,I)
  210 CONTINUE
C**** PRINT THE FIXED-END CORRECTIONS
      IF(NPAR(9).LT.1) GO TO 6695
      WRITE (6,4650)
      WRITE (6,4660) ((FEFC(I,J),J=1,5),I=1,12)
 6695 CONTINUE
C****
C
C     FORM THE TRANSFORMATION RELATING GLOBAL DISPLACEMENTS AND MEMBER
C     STRESS RESULTANTS AT NODES I AND J.
C
      DO 240 K1=1,10,3
      FAC = -1.0
      IF(K1.GT.4) FAC = 1.0
      NRS = K1-1
      DO 240 K2=1,10,3
      NCS = K2-1
      DO 230 IR=1,3
      NR = NRS+IR
      DO 230 IC=1,3
      NC = NCS+IC
      SA(NR,NC) = 0.0
      DO 220 IN=1,3
      N = NCS+IN
      SA(NR,NC) = SA(NR,NC) + FAC* S(NR,N)* DC(IC,IN)
  220 CONTINUE
  230 CONTINUE
```

```
  240 CONTINUE
C**** PRINT THE STRESS DISPLACEMENT TRANSFORMATION
      IF(NPAR(9).LT.1) GO TO 6696
      WRITE (6,4700)
      WRITE (6,4710) ((SA(I,J),J=1, 6),I=1,12)
      WRITE (6,4710) ((SA(I,J),J=7,12),I=1,12)
 6696 CONTINUE
C****
C
 4000 FORMAT (/// 24H NODE FLEXIBILITY MATRIX, // 1X)
 4010 FORMAT ( 1X / (6E20.8) )
 4020 FORMAT (/// 22H NODE STIFFNESS MATRIX, // 1X)
 4030 FORMAT ( 1X / (6E20.8) )
 4050 FORMAT (/// 42H FREE NODE DISPLACEMENTS  (5 MEMBER LOADS), // 1X)
 4060 FORMAT (1X / (5E20.8) )
 4500 FORMAT (23H1LOCAL STIFFNESS MATRIX, // 1X)
 4510 FORMAT (// (/6E15.6) )
 4600 FORMAT (// 17H0FIXED END FORCES, // 1X)
 4610 FORMAT (5E20.8)
 4650 FORMAT (// 43H0STRESS CORRECTIONS DUE TO FIXED END FORCES, // 1X)
 4660 FORMAT (5E20.8)
 4700 FORMAT (//35H0STRESS-DISPLACEMENT TRANSFORMATION, / 1X)
 4710 FORMAT (/// (6E20.8) )
C
      RETURN
      END
      SUBROUTINE TDCOS (N1,N2,N3,X,Y,Z,A,B)
C
C     CALLED BY?  STRETR,QTSHEL
C
C     THIS SUBROUTINE COMPUTES THE DIRECTION COSINES OF THE LOCAL
C     SYSTEM AND THE PROJECTED DIMENSIONS OF A TRIANGLE COMPONENT
C
      COMMON /TRANSF/  T1(3),T2(3),T3(3), T(9)
      EQUIVALENCE  (T11,T1(1)),(T12,T1(2)),(T13,T1(3)),(T21,T2(1)),
     1  (T22,T2(2)),(T23,T2(3)),(T31,T3(1)),(T32,T3(2)),(T33,T3(3))
      DIMENSION  X(1), Y(1), Z(1), A(1), B(1)
      A1 = X(N1)-X(N3)
      B1 = Y(N1)-Y(N3)
      C1 = Z(N1)-Z(N3)
      A2 = X(N2)-X(N3)
      B2 = Y(N2)-Y(N3)
      C2 = Z(N2)-Z(N3)
      T31 = B1*C2-B2*C1
      T32 = C1*A2-C2*A1
      T33 = A1*B2-A2*B1
      S =SQRT (T31**2+T32**2+T33**2)
      T31 = T31/S
      T32 = T32/S
      T33 = T33/S
      T11 = T33*T(5)-T32*T(8)
      T12 = T31*T(8)-T33*T(2)
      T13 = T32*T(2)-T31*T(5)
      S =SQRT (T11**2+T12**2+T13**2)
      T11 = T11/S
```

```
      T12 = T12/S
      T13 = T13/S
      T21 = T13*T32-T12*T33
      T22 = T11*T33-T13*T31
      T23 = T12*T31-T11*T32
      A(1)  = -T11*A2-T12*B2-T13*C2
      A(2)  =  T11*A1+T12*B1+T13*C1
      B(1)  =  T21*A2+T22*B2+T23*C2
      B(2)  = -T21*A1-T22*B1-T23*C1
      A(3)  = -A(1)-A(2)
      B(3)  = -B(1)-B(2)
      RETURN
      END
      SUBROUTINE THDFE (ID,X,Y,Z,T,DEN,RHO,NTP,EE,
     1          DCA,NFACE,LT,PWA,LOC,MAXPTS,SS,
     2          NUME,NUMMAT,MAXTP,NORTHO,NDLS,MAXNOD,
     3          NOPSET,INTRS,INTT,NUMNP)
C
C     CALLED BY ? SOL21
C     CALLS ? INP21,CALBAN,SSLAW,DER3DS,ST8R21,FACEPR
C
C
C     ROUTINE FOR THE STIFFNESS, MASS AND STRESS MATRIX GENERATION
C     FOR THE 8-TO-21 NODE ISO-(OR SUB)-PARAMETRIC ORTHOTROPIC
C     HEXAHEDRON.
C
      COMMON /JUNK/ XLF(4),YLF(4),ZLF(4),TLF(4),PLF(4),FILL1(22),V2(3),
     1          FILL2(12),LS(4),KLS(4),NOD(21),NOD9M(13),KOD(21),
     2          NREAD,TAG,E(12)
      COMMON /ELPAR/IFILL3(15),MBAND
      COMMON /EM/ SDT(42,63),SF(42,4),NS,ND,LM(63)
      DIMENSION RF(63,4),XM(63),D(6,6),TEMP(6,6),DUM(6,6),
     *          ALPHA(6),XX(3,21),B(6,63),H(21),P(3,21),SIGDT(6),
     *          DELT(21),FT(63),DL(21),PL(63),LOCOP(7),VIS(6)
C
      COMMON /GAUSS/ XG(4,4),WGT(4,4),STPTS(27,3)
      COMMON /DYN / IFILL4(11),NDYN
      COMMON /EXTRA/ MODEX,NT8
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      DIMENSION ID(NUMNP,1),X(1),Y(1),Z(1),T(1),DEN(1),RHO(1),
     1          NTP(1),EE(MAXTP,13,1),DCA(3,3,1),NFACE(1),LT(1),
     2          PWA(7,1),LOC(7,1),MAXPTS(1),SS(63,1)
C
C
      DATA  TG1, TG2  /'*', ' '/
      STPTS(1,1)=1.
      STPTS(2,1)=-1.
      STPTS(3,1)=-1.
      STPTS(4,1)=1.
      STPTS(5,1)=1.
      STPTS(6,1)=-1.
      STPTS(7,1)=-1.
      STPTS(8,1)=1.
```

```
STPTS(9,1)=0.
STPTS(10,1)=-1.
STPTS(11,1)=0.
STPTS(12,1)=1.
STPTS(13,1)=0.
STPTS(14,1)=-1.
STPTS(15,1)=0.
STPTS(16,1)=1.
STPTS(17,1)=1.
STPTS(18,1)=-1.
STPTS(19,1)=-1.
STPTS(20,1)=1.
STPTS(21,1)=0.
STPTS(22,1)=1.
STPTS(23,1)=-1.
STPTS(24,1)=0.
STPTS(25,1)=0.
STPTS(26,1)=0.
STPTS(27,1)=0.
STPTS(1,2)=1.
STPTS(2,2)=1.
STPTS(3,2)=-1.
STPTS(4,2)=-1.
STPTS(5,2)=1.
STPTS(6,2)=1.
STPTS(7,2)=-1.
STPTS(8,2)=-1.
STPTS(9,2)=1.
STPTS(10,2)=0.
STPTS(11,2)=-1.
STPTS(12,2)=0.
STPTS(13,2)=1.
STPTS(14,2)=0.
STPTS(15,2)=-1.
STPTS(16,2)=0.
STPTS(17,2)=1.
STPTS(18,2)=1.
STPTS(19,2)=-1.
STPTS(20,2)=-1.
STPTS(21,2)=0.
STPTS(22,2)=0.
STPTS(23,2)=0.
STPTS(24,2)=1.
STPTS(25,2)=-1
STPTS(26,2)=0.
STPTS(27,2)=0.
STPTS( 1,3)=1.
STPTS( 2,3)=1.
STPTS( 3,3)=1.
STPTS( 4,3)=1.
STPTS( 5,3)=-1.
STPTS( 6,3)=-1.
STPTS( 7,3)=-1.
STPTS( 8,3)=-1.
STPTS( 9,3)= 1.
```

```
      STPTS(10,3)= 1.
      STPTS(11,3)= 1.
      STPTS(12,3)= 1.
      STPTS(13,3)=-1.
      STPTS(14,3)=-1.
      STPTS(15,3)=-1.
      STPTS(16,3)=-1.
      STPTS(17,3)=0.
      STPTS(18,3)=0.
      STPTS(19,3)=0.
      STPTS(20,3)=0.
      STPTS(21,3)=0.
      STPTS(22,3)=0.
      STPTS(23,3)=0.
      STPTS(24,3)=0.
      STPTS(25,3)=0.
      STPTS(26,3)=1.
      STPTS(27,3)=-1.
      XG(1,1)  = 0.
      XG(2,1)  = 0.
      XG(3,1)  = 0.
      XG(4,1)  = 0.
      XG(1,2)  =    -.577350269189600
      XG(2,2)  =     .577350269189600
      XG(3,2)  = 0.
      XG(4,2)  = 0.
      XG(1,3)  =    -.774596669241500
      XG(2,3)  = 0.
      XG(3,3)  =     .774596669241500
      XG(4,3)  = 0.
      XG(1,4)  =    -.861136311594100
      XG(2,4)  =    -.339981043584900
      XG(3,4)  =     .339981043584900
      XG(4,4)  =     .861136311594100
      WGT(1,1)  = 2.0
      WGT(2,1)  = 0.0
      WGT(3,1)  = 0.0
      WGT(4,1)  = 0.0
      WGT(1,2)  = 1.0
      WGT(2,2)  = 1.0
      WGT(3,2)  = 0.0
      WGT(4,2)  = 0.0
      WGT(1,3)  = .5555555555556    D0
      WGT(2,3)  = .8888888888889    D0
      WGT(3,3)  = .5555555555556    D0
      WGT(4,3)  = 0.0
      WGT(1,4)  = .3478548451375    D0
      WGT(2,4)  = .6521451548625    D0
      WGT(3,4)  = .6521451548625    D0
      WGT(4,4)  = .3478548451375    D0
C
      NT8SV = MODEX
      DO 10 I=4,6
      DO 10 J=1,63
   10 B(I,J) = 0.0
```

```
      DO 14 I=1,42
      DO 14 J=1,4
   14 SF(I,J)=0.0
C
C     PRINT ELEMENT CONTROL VARIABLES
C
      WRITE (6,3001) NUME,NUMMAT,MAXTP,NORTHO,NDLS,MAXNOD,NOPSET,INTRS,
     1          INTT
C
C     READ AND CHECK INPUT UP TO THE ELEMENT DATA CARDS
C
      CALL INP21      (NUMMAT,MAXTP,NORTHO,NDLS,NOPSET,NT8SV,NUMNP,X,
     1                  Y,Z,DEN,RHO,NTP,EE,DCA,NFACE,LT,PWA,LOC,MAXPTS)
C
C     READ ELEMENT DATA CARDS
C
                  NREAD = 8
      IF(MAXNOD.GT.8) NREAD = 21
C
      WRITE (6,3014) (I,I=1,8)
      IF(MAXNOD.GT.8)
     *WRITE (6,3016) (I,I=9,21)
C
      NEL = 0
C
C     CARD FOR ELEMENT NUMBER ONE ONLY
C
      READ (5,1008) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(I),I=1,4)
      READ (5,1009) (NOD(I),I=1,NREAD)
      IREUSE = 0
      IF(INEL.EQ.1) GO TO 51
      WRITE (6,4014) INEL
      WRITE (6,4014)
      STOP
C
C     CARDS FOR ALL OTHER ELEMENTS
C
   50 READ (5,1008) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(I),I=1,4)
      READ (5,1009) (NOD(I),I=1,NREAD)
C
C     DATA ADMISSIBILITY CHECK
C
   51 IF(NDIS.EQ.0) NDIS = MAXNOD
      IF(NDIS.LE.MAXNOD) GO TO 5051
      WRITE (6,3015) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(I),I=1,4)
      WRITE (6,4015) NDIS,MAXNOD
      STOP
 5051 IF(NDIS.GE.8) GO TO 52
      WRITE (6,4023) NDIS
      STOP
   52 IF(NXYZ.EQ.0) NXYZ = NDIS
      IF(NXYZ.LE.NDIS) GO TO 5052
```

```
      WRITE (6,4016) NXYZ,NDIS
      WRITE (6,4099)
      MODEX = 1
      GO TO 53
 5052 IF(NXYZ.GE.8) GO TO 53
      WRITE (6,4024) NXYZ
      WRITE (6,4099)
      MODEX = 1
   53 IF(NMAT.GE.1 .AND. NMAT.LE.NUMMAT) GO TO 54
      WRITE (6,3015) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(I),I=1,4)
      WRITE (6,4017)
      WRITE (6,4099)
      MODEX = 1
   54 IF(MAXES.LE.NORTHO) GO TO 55
      WRITE (6,3015) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(I),I=1,4)
      WRITE (6,4018)
      WRITE (6,4099)
      MODEX = 1
   55 IF(IOP.GE.0 .AND. IOP.LE.NOPSET) GO TO 56
      WRITE (6,3015) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(I),I=1,4)
      WRITE (6,4019)
      WRITE (6,4099)
      MODEX = 1
   56 DO 57 I=1,4
      IF(LS(I).GE.0 .AND. LS(I).LE.NDLS) GO TO 57
      WRITE (6,3015) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(J),J=1,4)
      WRITE (6,4020) LS(I)
      WRITE (6,4099)
      MODEX = 1
   57 CONTINUE
C
C     DEFAULT VALUES IF REQUIRED
C
      IF(KG.EQ.0) KG = 1
      IF(NRSINT.EQ.0) NRSINT = INTRS
      IF(NTINT.EQ.0) NTINT = INTT
C
      DO 58 I=1,8
      IF(NOD(I).GE.1 .AND. NOD(I).LE.NUMNP) GO TO 58
      WRITE (6,3015) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(J),J=1,4)
      WRITE (6,4021) I,NOD(I)
      STOP
   58 CONTINUE
      IF(MAXNOD.LT.9) GO TO 60
      II = 0
      DO 59 I=9,21
      IF(NOD(I).EQ.0) GO TO 59
      II = II + 1
      NOD9M(II) = I
      IF(NOD(I).LE.NUMNP) GO TO 59
```

```
      WRITE (6,3015) INEL,NDIS,NXYZ,NMAT,MAXES,IOP,TZ,KG,NRSINT,NTINT
     1,IREUSE,(LS(J),J=1,4)
      WRITE (6,4021) I,NOD(I)
      STOP
   59 CONTINUE
C
      I = II + 8
      IF(I.EQ.NDIS) GO TO 60
      WRITE (6,4025) I,NDIS
      STOP
C
   60 NEL = NEL + 1
      ML = INEL - NEL
      IF(ML) 65,70,80
   65 WRITE (6,4022) INEL
      STOP
C
C     SAVE THE DATA FOR ELEMENT NUMBER *INEL* FOR POSSIBLE USE IN
C     DATA GENERATION
C
C
   70 KDIS = NDIS
      KXYZ  = NXYZ
      KMAT  = NMAT
      KAXES = MAXES
      KIOP  = IOP
      TTZ   = TZ
      KKG   = KG
      KRSINT = NRSINT
      KTINT  = NTINT
      KREUSE = IREUSE
      DO 72 I=1,4
   72 KLS(I) = LS(I)
      DO 74 I=1,NREAD
   74 KOD(I) = NOD(I)
      TAG = TG1
C
      GO TO 90
C
C     INCREMENT THE NON-ZERO NODE NUMBERS FROM THE PRECEEDING ELEMENT
C
   80 DO 85 I=1,NREAD
      IF(KOD(I).LT.1) GO TO 85
      KOD(I) = KOD(I) + KKG
   85 CONTINUE
      TAG = TG2
C
   90 ND = 3 * KDIS
C
C     COMPUTE THE AVERAGE ELEMENT TEMPERATURE USING COORDINATE NODES
C
      TAV = 0.0
      DO 95 K=1,KXYZ
      I = KOD(K)
   95 TAV = TAV + T(I)
```

```
      TAV = TAV / KXYZ
C
C     PERFORM TEMPERATURE INTERPOLATION FOR THE PROPERTY SET
C
      NT = NTP(KMAT)
      IF(NT.GT.1) GO TO 100
   97 DO 98 I=1,12
   98 E(I) = EE(1,I+1,KMAT)
      GO TO 112
  100 IF(TAV.GE.EE(1,1,KMAT)) GO TO 104
  102 WRITE (6,4030) TAV,NEL,KMAT
      STOP
  104 IF(TAV.GT.EE(NT,1,KMAT)) GO TO 102
      IF(TAV.EQ.EE(1,1,KMAT)) GO TO 97
C
      IF(MODEX.EQ.1) GO TO 112
C
      DO 106 K=2,NT
      K2 = K
      K1 = K-1
      IF(TAV.GT.EE(K1,1,KMAT) .AND. TAV.LE.EE(K2,1,KMAT)) GO TO 108
  106 CONTINUE
  108 DT = EE(K2,1,KMAT) - EE(K1,1,KMAT)
      RATIO = (TAV - EE(K1,1,KMAT)) / DT
      DO 110 I=1,12
  110 E(I) = EE(K1,I+1,KMAT) + RATIO *(EE(K2,I+1,KMAT)-EE(K1,I+1,KMAT))
C
  112 CONTINUE
C
C     FORM THE STRESS-STRAIN LAW IN MATERIAL COORDINATES AND TRANSFORM
C     TO GLOBAL (X,Y,Z) COORDINATES
C
      IF(MODEX.EQ.0)
     *CALL SSLAW (D,E,TEMP,DCA(1,1,KAXES),KAXES,KMAT,NEL,DUM,ALPHA)
C
C     STORE THE NODE COORDINATES FOR THIS ELEMENT
C
      IF(MODEX.EQ.1) GO TO 410
C
      DO 130 I=1,KDIS
      II = KOD(I)
      IF(I.LT.9) GO TO 125
      JJ = NOD9M(I-8)
      II = KOD(JJ)
  125 XX(1,I) = X(II)
      XX(2,I) = Y(II)
      XX(3,I) = Z(II)
  130 CONTINUE
C
C     COMPUTE THE ELEMENT STIFFNESS, MASS, THERMAL AND GRAVITY LOAD
C     MATRICES
C
      DO 170 I=1,63
      DO 170 J=1,4
  170 RF(I,J)=0.0
```

```
C
      IF(KREUSE.EQ.1) GO TO 300
C
      DO 180 I=1,KDIS
  180 DL(I)=0.0
      DO 190 I=1,ND
C
C
C         1. THERMAL LOADS
C
  190 FT(I)=0.0
      KTL = 0
      DUX = 0.0
      DO 200 I=1,4
  200 DUX = DUX +ABS(TLF(I))
      IF(DUX.GT.1.0E-06) KTL = 1
      IF  (NDYN.GT.0) KTL=0
      IF(KTL.EQ.0 .OR. NDYN.GT.0) GO TO 235
C
C           A.INITIAL STRESS CONSTANTS
C
      DO 210 I=1,6
      SIGDT(I) = 0.0
      DO 205 K=1,6
  205 SIGDT(I) = SIGDT(I) + D(I,K)* ALPHA(K)
  210 CONTINUE
C
C           B. VECTOR OF NODE TEMPERATURE DIFFERENCES
C
      DO 230 I=1,KDIS
      II = KOD(I)
      IF(I.LT.9) GO TO 220
      J = NOD9M(I-8)
      II = KOD(J)
  220 DELT(I) = T(II) - TTZ
  230 CONTINUE
C
C           C. CLEAR THE THERMAL LOAD NODE FORCE VECTOR
C
C         2. GRAVITY LOADS
C
  235 DUX=0.0
      DO 250 I=1,4
  250 DUX = DUX +ABS(XLF(I)) +ABS(YLF(I)) +ABS(ZLF(I))
      KGL = 0
      IF(DUX.GT.1.0E-6) KGL = 1
      IF  (NDYN.GT.0) KGL=0
C
C
C         3. MASS MATRIX
      KMS = 0
      IF(NDYN.GT.0) KMS = 1
C
      DO 270 K=1,ND
C
```

```
C        4. STIFFNESS MATRIX
C
  270 XM(K) = 0.0
      DO 280 I=1,ND
      DO 280 K=1,ND
  280 SS(I,K) = 0.0
C
C
      CALL ST8R21 (D,B,SS,XX,NOD9M,H,P,SIGDT,DELT,FT,DL,XM,NEL,ND,KDIS,
     1         KXYZ,KTL,KGL,KMS,KRSINT,KTINT,DEN(KMAT),RHO(KMAT))
C
C
C     NODE FORCES DUE TO THERMAL DISTORTION
C
  300 IF (KTL.EQ.0) GO TO 325
      DO 320 I=1,ND
      DO 310 K=1,4
  310 RF(I,K) = FT(I)* TLF(K)
  320 CONTINUE
C
C     NODE FORCES DUE TO STATIC ACCELERATIONS
C
C
  325 IF (KGL.EQ.0) GO TO 350
      DO 340 I=1,KDIS
      K3 = 3*I
      K2 = K3-1
      K1 = K2-1
      DO 330 L=1,4
      RF(K1,L) = RF(K1,L) + XLF(L)*DL(I)
      RF(K2,L) = RF(K2,L) + YLF(L)* DL(I)
  330 RF(K3,L) = RF(K3,L) + ZLF(L)* DL(I)
  340 CONTINUE
C
C     COMPUTE NODE FORCES DUE TO ELEMENT SURFACE LOADINGS
C
  350 IF(NDLS.LT.1.OR.NDYN.GT.0) GO TO 405
C
      DO 400 L=1,4
      IF(PLF(L).EQ.0.0) GO TO 400
      M = KLS(L)
      IF(M.LT.1) GO TO 400
      DO 360 K=1,ND
C
  360 PL(K) = 0.0
      CALL FACEPR (NEL,KDIS,KXYZ,XX,NOD9M,H,P,PL,NFACE(M),LT(M),
     1          PWA(1,M),M)
C
      DO 370 I=1,ND
C
  370 RF(I,L) = RF(I,L) + PL(I)* PLF(L)
  400 CONTINUE
  405 CONTINUE
C
C     ASSIGN EQUATION NUMBERS TO THE ELEMENT DEGREES OF FREEDOM
```

```
C
  410 K = -3
      DO 420 I=1,KDIS
      II = KOD(I)
      IF(I.LT.9) GO TO 415
      JJ = NOD9M(I-8)
      II = KOD(JJ)
  415 K = K+3
      DO 420 L=1,3
      M = K+L
  420 LM(M) = ID(II,L)
C
      IF(KIOP.GT.0) NS = 6*MAXPTS(KIOP)
      IF(KIOP.EQ.0) NS = 6
      IF  (NDYN.GT.0) NS=42
C
C     SAVE STIFFNESS AND LOAD MATRICES
C
      CALL CALBAN (MBAND,NDIF,LM,XM,SS,RF,ND,63,NS)
C
C     COMPUTE STRESS RECOVERY MATRICES
C
      IF  (NDYN.LT.1) GO TO 425
      NOP=7
      DO 422 I=1,7
  422 LOCOP(I)=I + 20
      GO TO 450
  425 IF  (KIOP.EQ.0) GO TO 440
      NOP = MAXPTS(KIOP)
      DO 430 I=1,NOP
  430 LOCOP(I) = LOC(I,KIOP)
      GO TO 450
  440 NOP = 1
      LOCOP(1) = 21
C
  450 IF(MODEX.EQ.1) GO TO 510
C
C     CONSIDER EACH OUTPUT LOCATION
C
      DO 500 L=1,NOP
C
      M= LOCOP(L)
      E1= STPTS(M,1)
      E2= STPTS(M,2)
      E3= STPTS(M,3)
C
C     COMPUTE THE STRAIN-DISPLACEMENT MATRIX AT THIS LOCATION
C
      CALL DER3DS (MEL,XX,B,DET,E1,E2,E3,NOD9M,H,P,KDIS,KXYZ)
C
      DO 470 I=1,6
      N= 6*(L-1)+I
      DO 465 J=1,ND
      Q = 0.0
      DO 460 K=1,6
```

```
  460 Q = Q + D(I,K)* B(K,J)
  465 SDT(N,J)  = Q
  470 CONTINUE
C
C     FORM THE INITIAL STRESS CORRECTIONS DUE TO THERMAL LOADS
C
      IF(KTL.EQ.0 .OR. NDYN.GT.0) GO TO 500
C
C
C        1. TEMPERATURE DIFFERENCE AT THIS LOCATION
C
      Q = 0.0
      DO 480 K=1,KDIS
C
C        2. VECTOR OF INITIAL STRESSES
C
  480 Q = Q + H(K)* DELT(K)
      DO 485 K=1,6
  485 VIS(K)  = -Q * SIGDT(K)
C
      DO 490 I=1,6
      N = 6*(L-1)+I
C
      DO 490 K=1,4
  490 SF(N,K)  = VIS(I)* TLF(K)
C
  500 CONTINUE
C
C     SAVE THE STRESS RECOVERY ARRAYS
C
C
  510 CONTINUE
C
      IF(MODEX.EQ.0)
     1WRITE (1) ND,NS,(LM(I),I=1,ND),((SDT(I,J),I=1,NS),J=1,ND),
     2          ((SF(I,J),I=1,NS),J=1,4)
C
C     PRINT DATA FOR THE CURRENT ELEMENT
C
      WRITE (6,3015) NEL,KDIS,KXYZ,KMAT,KAXES,KIOP,TTZ,KKG,KRSINT,KTINT,
     1          KREUSE,KLS
      WRITE (6,3017) (KOD(I),I=1,NREAD)
C
C***  DATA PORTHOLE SAVE
      IF(NT8SV.EQ.1)
     1WRITE (NT8)   NEL,KDIS,KXYZ,KMAT,KAXES,KIOP,TTZ,   KRSINT,KTINT,
     2          KREUSE,KLS,NREAD,
     3          (KOD(I),I=1,NREAD)
C***
C
C     CHECK FOR THE LAST ELEMENT
C
      IF(NUME-NEL) 65,600,530
  530 IF(ML)       50, 50, 60
C
```

```
  600 RETURN
C
C     FORMATS
C
 1008 FORMAT (6I5,F10.0,4I5,4I2)
 1009 FORMAT (16I5)
C
 3001 FORMAT ( 7X,34HNUMBER OF 21-NODE ELEMENTS       = I6//
     1         7X,34HNUMBER OF MATERIAL SETS          = I6//
     2         7X,26HMAXIMUM NUMBER OF MATERIAL,      /
     3         7X,34HTEMPERATURE INPUT POINTS         = I6 //
     4         7X,18HNUMBER OF MATERIAL,              /
     5         7X,34HAXIS ORIENTATION SETS            = I6//
     *         7X,34HNUMBER OF DISTRIBUTED LOAD SETS  = I6//
     6         7X,34HMAXIMUM NUMBER OF ELEMENT NODES  = I6 //
     7         7X,34HNUMBER OF STRESS OUTPUT SETS     = I6 //
     8         7X,34HR,S COORDINATE INTEGRATION ORDER = I6 //
     9         7X,34HT COORDINATE INTEGRATION ORDER   =I6 // 1X)
 3014 FORMAT (52H13 / D   8   T O   2 1   N O D E   S O L I D   E L E ,
     1  18H M E N T   D A T A, //  8H ELEMENT 2(2X,5HNODES),2(2X,
     2  5HMATL.),2X,6HSTRESS,4X,6HSTRESS,2X,4HNODE,2(2X,5HGAUSS),6X,
     3  2HK-,5X,3HLSA,3X,3HLSB,3X,3HLSC,3X,3HLSD,  /
     4  8H  NUMBER,7H -NDIS-,7H -NXYZ-,2X,5HTABLE,3X,4HAXES,2X,6HOUTPUT,
     5  6X,4HFREE,2X,4HINC.,2(3X,4HPTS.),2X,6HMATRIX,2X,4(2X,4H-OR-), /
     6  26X,3HNO.,4X,3HSET,5X,3HSET,5X,5HTEMP.,2X,4H-KG-,2X,5H-R,S-,4X,
     7  3H-T-,2X,6HRE-USE,2X, 8(2X,2HN-,I2)  )
 3015 FORMAT (I8,4I7,I8,F10.1,I6,2I7,I8,2X,4I6)
 3016 FORMAT (84X,8(2X,2HN-,I2) / 84X,5(2X,2HN-,I2) )
 3017 FORMAT (84X,8I6 / 84X,8I6,/ 84X,5I6)
C
 4014 FORMAT (33H0ERROR***  ENCOUNTERED ELEMENT (,I5,13H), BUT EXPECT,
     1        21H TO READ ELEMENT ONE., / 1X)
 4015 FORMAT (42H0ERROR***  NUMBER OF DISPLACEMENT NODES (,I5,4H) IS,
     1        30H LARGER THAN MAXIMUM ALLOWED (,I5,2H)., / 1X)
 4016 FORMAT (40H0ERROR***  NUMBER OF COORDINATE NODES (,I5,6H) MUST,
     1         39H BE .LE. NUMBER OF DISPLACEMENT NODES (,I5,2H).)
 4017 FORMAT (36H0ERROR***  ILLEGAL MATERIAL NUMBER. )
 4018 FORMAT (44H0ERROR***  ILLEGAL MATERIAL AXIS REFERENCE.  )
 4019 FORMAT (41H0ERROR***  ILLEGAL OUTPUT SET REFRENCE.  )
 4020 FORMAT (41H0ERROR***  PRESSURE LOAD SET REFERENCE (,I5,4H) IS,
     1         9H ILLEGAL. )
 4021 FORMAT (16H0ERROR***  THE ,I2,18H-TH ELEMENT NODE (,I5,4H) IS,
     1         9H ILLEGAL.,/ 1X)
 4022 FORMAT (28H0ERROR***  ELEMENT NUMBER (,I5,11H) IS OUT OF,
     1         10H SEQUENCE., / 1X)
 4023 FORMAT (42H0ERROR***  NUMBER OF DISPLACEMENT NODES (,I5,
     1         25H) MUST BE AT LEAST EIGHT. )
 4024 FORMAT (40H0ERROR***  NUMBER OF COORDINATE NODES (,I5,
     1         25H) MUST BE AT LEAST EIGHT. )
 4025 FORMAT (38H0ERROR***  NUMBER OF NON-ZERO NODES (,I3,6H) READ,
     1         50H DOES NOT EQUAL THE NUMBER OF DISPLACEMENT NODES (,
     2         I3,2H).,/ 1X)
 4030 FORMAT (33H0ERROR***  AVERAGE TEMPERATURE (,F10.2,5H) FOR,
     1         10H ELEMENT (,I5,29H) OUT OF RANGE FOR MATERIAL (,I3,
     2         2H)., / 1X)
```

```
 4099 FORMAT (12X,31HPROCEED IN DATA CHECK ONLY MODE, / 1X)
C
      END
      SUBROUTINE THREED
C
C
C     CALLS?  BRICK8,STRSC,PRIST
C     CALLED BY?  ELTYPE
C
      COMMON /ELPAR/ NPAR(14),NUMNP,MBAND,NELTYP,N1,N2,N3,N4,N5,MTOT,NEQ
      COMMON /EM/ NS,ND,B(42,63),TT(42,4),LM(63)
      EQUIVALENCE (IS1,TT(4)) , (IS2,TT(6))
      COMMON /JUNK/  LT,LH,L,IPAD,SIG(24),N6,N7,N8,N9,N10,N11,
     1               N12,IFILL(371)
      COMMON /EXTRA/ MODEX,NT8,N10SV,NT10,IFILL2(12)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
      COMMON /one/ A(1)
      DIMENSION SPR(6)
C
      IF(NPAR(1).EQ.0) GO TO 500
      N6=N5+NUMNP
      N7=N6+NPAR(3)
      N8=N7+NPAR(3)
      N9=N8+NPAR(3)
      N10=N9+NPAR(3)
      N11=N10+NPAR(4)
      N12=N11+NPAR(4)
      N13=N12+NPAR(4)
      N14=N13+NPAR(4)
      N15=N14+33*33
      N16=N15+12*33
      IF(N16.GT.MTOT) CALL ERROR (N16-MTOT)
C
      CALL BRICK8 (A(N14),A(N15),NPAR(2),NPAR(3),NPAR(4),A(N1),A(N2),
     1            A(N3),A(N4),A(N5),A(N6),A(N7),A(N8),A(N9),A(N10),
     2            A(N11),A(N12),A(N13),NUMNP)
C
      RETURN
C
  500 WRITE (6,2005)
      NUME=NPAR(2)
      DO 800 MM=1,NUME
      CALL STRSC (A(N1),A(N3),NEQ,0)
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) NS
      WRITE (6,2000)
      DO 800 L=LT,LH
      CALL STRSC (A(N1),A(N3),NEQ,1)
      CALL PRIST (NS,IS1,IS2,SIG,SPR)
      WRITE (6,3005) MM,L,IS1,(SIG(I),I=1,6),(SPR(I),I=1,3)
C***  STRESS PORTHOLE
      IF(N10SV.EQ.1)
     *WRITE (NT10) MM,L,IS1,(SIG(I),I=1,6),(SPR(I),I=1,3)
```

```
      IF(NIOSV.EQ.1 .AND. NS.EQ.12)
     *WRITE (NT10) IS2,(SIG(I),I=7,12),(SPR(I),I=4,6)
      IF(NS.EQ.12) WRITE (6,3015) IS2,(SIG(I),I=7,12),(SPR(I),I=4,6)
  800 CONTINUE
C
      RETURN
C
 2000 FORMAT (/)
 2005 FORMAT (36H1.....8-NODE SOLID ELEMENT STRESSES    //
     . 24H ELEMENT  LOAD NO.  FACE    ,5X,
     .    104H SIG-XX       SIG-YY       SIG-ZZ       SIG-XY       SIG-YZ
     .   SIG-ZX       SIG-MAX      SIG-MIN     S2/ANGLE)
 3005 FORMAT (I6,I9,I8,2X,1P9E12.2)
 3015 FORMAT (15X,  I8,2X,1P9E12.2)
      END
      SUBROUTINE TPLATE(NUMEL,NUMMAT,ID,X,Y,Z,C,NUMNP,MBAND)
C
C     CALLS?  QTSHEL,STRETR,CALBAN
C     CALLED BY?  SHELL
C
      COMMON/QTSARG/
     1XX(5),YY(5),ZZ(5),HM(5),HP(5),CM(3,3),ALFA(3),HW(5),RHO(5,3),P(5)
     2, T(5),DT(5),SM(5,3),BM(5,3),TDIS(36),TROT(36),S(30,30),R(30)
      COMMON/EM/LM(24),ND,NS,ASA(24,24),RF(24,4),XM(24),SA(12,24)
     1,SF(12,4),PF(24),IFILL1(3000)
      COMMON /EXTRA/ MODEX,NT8,IFILL2(14)
      DIMENSION X(1),Y(1),Z(1),ID(NUMNP,1),C(12,1),IX(7),IY(7),EL(4)
     1, TLO(5,4)
      common /say/ neqq,numee,loopur,nnblock,nterms,option
      common /what/ naxa(10000),irowl(10000),icolh(10000)
C
      LL    = 4
      NDM   = 24
      MTYPE = 6
      ISTOP = 0
      NS    = 6
C     DEGREES OF FREEDOM PER NODE
      NPF   = 6
C     MID-SIDE NODES
      MID   = 0
C     GLOBAL REFERENCE FOR DISPLACEMENTS/ROTATIONS
      IDIS  = 0
      IROT  = 0
C
      WRITE (6,2000) MTYPE,NUMEL,NUMMAT
C
C *** READ AND PRINT OF MATERIAL PROPERTIES
C
      WRITE (6,2001)
      DO 10 N=1,NUMMAT
      READ  (5,1000) K,(C(I,K),I=1,12)
   10 WRITE (6,2002) K,(C(I,K),I=3,12)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) ((C(I,N),I=1,12),N=1,NUMMAT)
```

```
C
C *** READ AND PRINT OF ELEMENT LOAD MULTIPLIERS
C
      READ  (5,1002)  ((TLO(I,J),J=1,4),I=1,5)
      WRITE (6,2006)
      WRITE (6,2007)  (J,(TLO(I,J),I=1,5),J=1,4)
C***  DATA PORTHOLE SAVE
      IF(MODEX.EQ.1)
     *WRITE (NT8) TLO
C
C *** READ AND PRINT OF ELEMENT DATA
C
      WRITE (6,2003)
      NN=0
  100 READ  (5,1001) MM,IY,EL
  110 NN=NN+1
      IF (MM-NN) 440,50,60
   50 DO 45 I=1,7
   45 IX(I)=IY(I)
      INCL=IY(7)
      IF (IY(6).EQ.0) IY(6)=1
      IM=IY(6)
      IF(INCL.EQ.0) INCL=1
      NNS=5
      IF (IY(5).EQ.0) NNS=4
      IF (IY(4).EQ.0) NNS=3
      RHOM=C(3,IM)
      ALFA(1)=C(4,IM)
      ALFA(2)=C(5,IM)
      ALFA(3)=C(6,IM)
      CM(1,1)=C(7,IM)
      CM(1,2)=C(8,IM)
      CM(1,3)=C(9,IM)
      CM(2,2)=C(10,IM)
      CM(2,3)=C(11,IM)
      CM(3,3)=C(12,IM)
      CM(2,1)=CM(1,2)
      CM(3,1)=CM(1,3)
      CM(3,2)=CM(2,3)
C
      DO 30 I=1,5
      HM(I)=EL(1)
      HP(I)=EL(1)
      HW(I)=EL(1)
   30 CONTINUE
      GO TO 70
C
   60 DO 65 I=1,NNS
   65 IX(I)=IX(I)+INCL
C
   70 DO 40 I=1,NNS
      P(I) = 0.0
      DO 72 K=1,3
      RHO(I,K) = 0.0
      SM(I,K) = 0.0
```

```
   72 BM(I,K) = 0.0
      J=IX(I)
      XX(I)=X(J)
      YY(I)=Y(J)
   40 ZZ(I)=Z(J)
C
C     FORM SHELL GLOBAL STIFFNESS MATRIX
C
C
C
      CALL QTSHEL (-1,NNS,NPF,MID,IDIS,IROT,ND,NTF)
C
C     CLEAR STRESS CORRECTION AND LOAD VECTOR MATRICES
C
      DO 520 L=1,LL
      DO 514 I=1,NS
  514 SF(I,L) = 0.0
      DO 516 J=1,ND
  516 RF(J,L) = 0.0
      IF(MODEX.EQ.1) GO TO 200
  520 CONTINUE
C
C     FORM ELEMENT MASS, STRESS/DISPLACEMENT AND UNIT NORMAL PRESSURE
C     FORCE MATRICES
C
      CALL STRETR (NNS,RHOM)
C
C     FORM LOAD VECTORS AND STRESS CORRECTION MATRICES
C
      DO 550 IL=1,LL
C
C     CHECK FOR NO ELEMENT LOADINGS
      DUM = 0.0
      DO 522 K=1,5
  522 DUM = DUM +ABS(TLO(K,IL))
      IF(DUM.LT.1.0E-8) GO TO 550
C
C     GENERATE ELEMENT LOADS (MECHANICAL)
C
      DO 524 I=1,NNS
      K = 6*(I-1)
      DO 523 J=1,3
      K = K+1
      RF(K,IL) = XM(K)* TLO(J+2,IL) + PF(K)* TLO(1,IL)* EL(2)
  523 CONTINUE
      T(I)      = TLO(2,IL)* EL(3)
      DT(I)     =-TLO(2,IL)* EL(4)
  524 CONTINUE
C
C     GENERATE ELEMENT LOADS (THERMAL)
C
      DUM =ABS(T(1)) +ABS(DT(1))
      IF(DUM.LT.1.0E-8) GO TO 550
      DO 525 I=1,NNS
      DO 525 K=1,3
```

```
      SM(I,K) = 0.0
  525 BM(I,K) = 0.0
C
      CALL QTSHEL (1,NNS,NPF,MID,IDIS,IROT,ND,NTF)
C
      DO 526 I=1,ND
  526 RF(I,IL) = RF(I,IL) + R(I)
      DO 527 J=1,30
  527 R(J) = 0.0
C     ELEMENT STRESS CORRECTION MATRICES
      DO 528 I=1,NNS
      DT(I) = -DT(I)
      DO 528 K=1,3
      SM(I,K) = 0.0
  528 BM(I,K) = 0.0
C
      CALL QTSHEL (2,NNS,NPF,MID,IDIS,IROT,ND,NTF)
C
C     AVERAGE NODAL STRESSES AT THE ELEMENT CENTROID
C
      DO 530 I=1,NNS
      DO 530 K=1,3
      SF(K  ,IL) = SF(K  ,IL) + SM(I,K)
  530 SF(K+3,IL) = SF(K+3,IL) + BM(I,K)
      DUM = 1.0/DFLOAT(NNS)
      DO 532 K=1,6
  532 SF(K,IL) = SF(K,IL)* DUM
C
  550 CONTINUE
      GO TO 210
C
C***  DATA PORTHOLE SAVE
  200 WRITE (NT8) NN,(IX(I),I=1,6),EL
  210 CONTINUE
      WRITE (6,2004) NN,(IX(I),I=1,6),EL
C
C *** FORM LM ARRAY AND COMPUTE BANDWIDTH
C
      L = MINO(NNS,4)
      DO 410 I=1,L
      J=NPF*I-NPF
      N=IX(I)
      DO 410 K=1,NPF
  410 LM(J+K)=ID(N,K)
      IF (MODEX .EQ. 1) ND=NPF*MINO(NNS,4)
      IF(MODEX.EQ.1) GO TO 224
      DO 222 I=1,ND
      DO 222 J=1,ND
      ASA(I,J) = S(I,J)
  222 ASA(J,I) = S(I,J)
  224 CONTINUE
C
      CALL CALBAN (MBAND,NDIF,LM,XM,ASA,RF,ND,NDM,NS)
      IF(MODEX.EQ.1) GO TO 500
C
```

```
      WRITE (1) ND,NS,(LM(I),I=1,ND),((SA(I,J),I=1,NS),J=1,ND),
     1 ((SF(I,J),I=1,NS),J=1,4)
      GO TO 500
  440 WRITE (6,2005) MM
      ISTOP=1
  500 IF (NN.LT.MM) GO TO 110
      IF (NN.EQ.NUMEL) RETURN
      IF (ISTOP.EQ.1) STOP
      GO TO 100
C
 2000 FORMAT (50H1T H I N   P L A T E / S H E L L   E L E M E N T S, //
     1          22H ELEMENT TYPE         =, I5 /
     2          22H NUMBER OF ELEMENTS   =, I5 /
     3          22H NUMBER OF MATERIALS =, I5 //// 1X)
 1000 FORMAT (I10,6F10.0/6F10.0)
 2001 FORMAT (24H MATERIAL PROPERTY TABLE, // 9H MATERIAL,8X,4HMASS,4X,
     1 7HTHERMAL,2X,9HEXPANSION,2X,12HCOEFFICIENTS,14X,3H/ /,2X,
     2 13HE L A S T I C,4X,17HC O N S T A N T S,2X,3H/ /, / 3X,6HNUMBER,
     3 5X,7HDENSITY,4X,8HALPHA(X),4X,8HALPHA(Y),4X,8HALPHA(Z),7X,
     4 5HC(XX),7X,5HC(XY),7X,5HC(XG),7X,5HC(YY),7X,5HC(YG),7X,5HG(XY),
     5  / 1X)
 2002 FORMAT (I9,1P10E12.3)
 1002 FORMAT (4F10.0)
 2006 FORMAT (30H1ELEMENT LOAD CASE MULTIPLIERS, // 13H ELEMENT LOAD,
     1 4X,8HPRESSURE,5X,7HTHERMAL,13X,2HX-,13X,2HY-,13X,2HZ-, /
     2 13H  CASE NUMBER,17X,7HEFFECTS, 3(3X,12HACCELERATION), / 1X)
 2007 FORMAT (6X,I1,6X,2F12.3,3F15.3)
 2003 FORMAT (32H1THIN  PLATE/SHELL  ELEMENT DATA, // 8H ELEMENT, 42X,
     1 8HMATERIAL,4X,7HAVERAGE,4X,6HNORMAL,2X,11HTEMPERATURE,5X,
     2 7HTHERMAL, / 8H  NUMBER,2X,6HNODE-I,2X,6HNODE-J,2X,6HNODE-K,2X,
     3 6HNODE-L,2X,6HNODE-O,4X,6HNUMBER,2X,9HTHICKNESS,2X,8HPRESSURE,
     4 3X,10HDIFFERENCE,4X,8HGRADIENT, / 1X)
 1001 FORMAT (8I5,4F10.0)
 2004 FORMAT (6I8,I10,F11.4,F10.1,F13.2,F12.3)
 2005 FORMAT (19H0CARD FOR ELEMENT (,I5,14H) IS IN ERROR., / 1X)
C
      END
      SUBROUTINE TRFPRD (M,NEN,Q1,Q2,Q3,P1,P2,P3)
C
C     CALLED BY?  STRETR,QTSHEL
C
C     THIS SUBROUTINE GENERATES THE TRANSFORMATIONS RELATING A LOCAL
C     SUBTRIANGLE SYSTEM TO THE NODAL DIS/ROT SYSTEMS AT ITS 3 CORNERS
C
      COMMON /TRANSF/  T1(3),T2(3),T3(3), T(9)
      DIMENSION  P1(1), P2(1), P3(1), Q1(1), Q2(1), Q3(1)
      EQUIVALENCE (T11,T1(1)),(T12,T1(2)),(T13,T1(3)),(T21,T2(1)),
     1  (T22,T2(2)),(T23,T2(3)),(T31,T3(1)),(T32,T3(2)),(T33,T3(3))
      DO 300  I = 1,3
      J = I + 3
      K = I + 6
      P1(I)  = T1(I)
      P1(J)  = T1(I)
      P2(I)  = T2(I)
      P2(J)  = T2(I)
```

```
      P3(I) = T3(I)
      P3(J) = T3(I)
      IF (NEN.NE.4)  GO TO 150
      CI = T(I)
      CJ = T(J)
      CK = T(K)
      IF (M)  260,260,240
  150 IF (M)  180,180,200
  180 P1(K) = T1(I)
      P2(K) = T2(I)
      P3(K) = T3(I)
      GO TO 300
  200 CI = Q3(I)
      CJ = Q3(J)
      CK = Q3(K)
  240 P1(I) = T11*Q1(I) + T12*Q1(J) + T13*Q1(K)
      P1(J) = T11*Q2(I) + T12*Q2(J) + T13*Q2(K)
      P2(I) = T21*Q1(I) + T22*Q1(J) + T23*Q1(K)
      P2(J) = T21*Q2(I) + T22*Q2(J) + T23*Q2(K)
      P3(I) = T31*Q1(I) + T32*Q1(J) + T33*Q1(K)
      P3(J) = T31*Q2(I) + T32*Q2(J) + T33*Q2(K)
  260 P1(K) = T11*CI    + T12*CJ    + T13*CK
      P2(K) = T21*CI    + T22*CJ    + T23*CK
      P3(K) = T31*CI    + T32*CJ    + T33*CK
  300 CONTINUE
      RETURN
      END
      SUBROUTINE TRIFAC (A,B,MAXA,NEQB,MA,NBLOCK,NWA,NTB,NEQ,MI)
C
C     CALLED BY?  STEP
C
C     THIS ROUTINE DECOMPOSES THE SYSTEM MATRIX IN BLOCKS
C
      DIMENSION      A(NWA),B(NWA),MAXA(MI)
C
      COMMON /TAPES/ NSTIF,NRED,NL,NR,IFILL(2)
C
      MA2=MA - 2
      IF(MA2.EQ.O) MA2 = 1
      INC=NEQB - 1
C
C     SET TAPE ASSIGNMENTS
C
      NSTIF = 4
      NRED  = 3
      NL    = 1
      NR    = 7
C
      N1=NL
      N2=NR
      REWIND NSTIF
      REWIND NRED
      REWIND N1
      REWIND N2
C
```

```
C       MAIN LOOP OVER ALL BLOCKS
        DO 600 NJ=1,NBLOCK
        IF (NJ.NE.1) GO TO 10
        READ (NSTIF) A
        GO TO 100
 10     IF (NTB.EQ.1) GO TO 100
        REWIND N1
        REWIND N2
        READ (N1) A
C
C       FIND COLUMN HEIGHTS
 100    KU=1
        KM=MINO(MA,NEQB)
        MAXA(1)=1
        DO 110 N=2,MI
        IF (N-MA) 120,120,130
 120    KU=KU + NEQB
        KK=KU
       MM = MINO(N,KM)
        GO TO 140
 130    KU=KU + 1
        KK=KU
        IF (N-NEQB) 140,140,136
 136    MM=MM - 1
 140    DO 160 K=1,MM
        IF (A(KK)) 110,160,110
 160    KK=KK - INC
 110    MAXA(N)=KK
       IF(A(1)) 172,174,176
 174  KK = (NJ-1)*NEQB +1
       IF(KK.GT.NEQ) GO TO 590
       WRITE (6,1000) KK
       STOP
 172  KK = (NJ-1)*NEQB +1
       WRITE (6,1010) KK
C
C       FACTORIZE LEADING BLOCK
 176    DO 200 N=2,NEQB
        NH=MAXA(N)
        IF (NH-N) 200,200,210
 210    KL=N + INC
        KU=NH
        K=N
        D=0.
        DO 220 KK=KL,KU,INC
        K=K - 1
        C=A(KK)/A(K)
        D=D + C*A(KK)
 220    A(KK)=C
        A(N)=A(N) - D
C
        IF (A(N)) 222,224,230
 224    KK=(NJ-1)*NEQB + N
        IF (KK.GT.NEQ) GO TO 590
       WRITE (6,1000) KK
```

```
      STOP
  222 KK = (NJ-1)*NEQB +N
      WRITE (6,1010) KK
C
 230  IC=NEQB
      DO 240 J=1,MA2
      MJ=MAXA(N+J) - IC
      IF (MJ-N) 240,240,280
 280  KU=MIN0(MJ,NH)
      KN=N + IC
      C=0.
      DO 300 KK=KL,KU,INC
 300  C=C + A(KK)*A(KK+IC)
      A(KN)=A(KN) - C
 240  IC=IC + NEQB
C
 200  CONTINUE
C
C     CARRY OVER INTO TRAILING BLOCKS
 320  DO 400 NK=1,NTB
      IF ((NK+NJ).GT.NBLOCK) GO TO 400
      NI=N1
      IF ((NJ.EQ.1).OR.(NK.EQ.NTB)) NI=NSTIF
      READ (NI) B
      ML=NK*NEQB + 1
      MR=MIN0((NK+1)*NEQB,MI)
      MD = MI-ML
      KL=NEQB + (NK-1)*NEQB*NEQB
      N=1
C
      DO 500 M=ML,MR
      NH=MAXA(M)
      KL=KL + NEQB
      IF(NH-KL) 505,510,510
 510  KU=NH
      K=NEQB
      D=0.
      DO 520 KK=KL,KU,INC
      C=A(KK)/A(K)
      D=D + C*A(KK)
      A(KK)=C
 520  K=K - 1
      B(N)=B(N) - D
      IF (MD) 500,500,530
 530  IC=NEQB
      DO 540 J=1,MD
      MJ=MAXA(M+J) - IC
      IF (MJ-KL) 540,550,550
 550  KU=MIN0(MJ,NH)
      KN=N + IC
      C=0.
      DO 575 KK=KL,KU,INC
 575  C=C + A(KK)*A(KK+IC)
      B(KN)=B(KN) - C
 540  IC=IC + NEQB
```

```
  505 MD = MD-1
C
 500    N=N + 1
C
        IF (NTB.NE.1) GO TO 560
        WRITE (NRED) A,MAXA
        DO 570 I=1,NWA
 570    A(I)=B(I)
        GO TO 600
 560    WRITE (N2) B
C
 400    CONTINUE
C
        M=N1
        N1=N2
        N2=M
 590    WRITE (NRED) A,MAXA
C
 600    CONTINUE
C
 1000 FORMAT (44HOSTOP.  ZERO PIVOT ENCOUNTERED AT EQUATION (,I5,1H) )
 1010 FORMAT (52HOWARNING.   NEGATIVE PIVOT ENCOUNTERED DURING MATRIX,
     1         35H DECOMPOSITION AT EQUATION NUMBER (,I5,1H), 1X)
C
       RETURN
       END
      SUBROUTINE VECTOR(V,XI,YI,ZI,XJ,YJ,ZJ)
C
C     CALLED BY?  PLNAX,QUAD
C
      DIMENSION V(4)
      X=XJ-XI
      Y=YJ-YI
      Z=ZJ-ZI
      V(4)=SQRT(X*X+Y*Y+Z*Z)
C
      V(3)=Z/V(4)
      V(2)=Y/V(4)
      V(1)=X/V(4)
      RETURN
      END
      SUBROUTINE VECTR2 (V,XI,YI,ZI,XJ,YJ,ZJ,IERR)
C
C     CALLED BY ? INP21
C
C
C     THIS ROUTINE FORMS A UNIT LENGTH VECTOR *V* FROM POINT *I*
C     TO POINT *J* IN X,Y,Z SPACE
C
      DIMENSION V(3)
C
      IERR = 1
      X = XJ - XI
      Y = YJ - YI
      Z = ZJ - ZI
```

```
      XLN =SQRT(X*X+Y*Y+Z*Z)
      IF(XLN.LE.1.0E-08) RETURN
      XLN = 1.0 / XLN
      IERR = 0
      V(3) = Z * XLN
      V(2) = Y * XLN
      V(1) = X * XLN
      RETURN
      END
```